

AD-A224 326

LE COPY

AD _____

1

CONTRACT NO: DAMD17-87-C-7195

TITLE: THE PERSONAL MONITOR AND COMMUNICATOR (PMC)

SUBTITLE: An Electronic Dogtag

PRINCIPAL INVESTIGATOR: W. A. Tacker, Jr., M.D., Ph.D.

CONTRACTING ORGANIZATION: Purdue Research Foundation
West Lafayette, IN 47907

REPORT DATE: December 31, 1989

TYPE OF REPORT: Final Report

PREPARED FOR: U.S. ARMY MEDICAL RESEARCH AND DEVELOPMENT COMMAND
Port Detrick, Frederick, Maryland 21701-5012

DISTRIBUTION STATEMENT: Approved for public release;
distribution unlimited

The findings in this report are not to be construed as an
official Department of the Army position unless so designated by
other authorized documents.

20030206098

90 06 27 116

DTIC
ELECTE
JUN 28 1990
S D

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Purdue Research Foundation		6b. OFFICE SYMBOL (If applicable)		7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code) West Lafayette, IN 47907		7b. ADDRESS (City, State, and ZIP Code)			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Medical Research & Development Command		8b. OFFICE SYMBOL (If applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract No. DAMD17-87-C-7195	
8c. ADDRESS (City, State, and ZIP Code) Fort Detrick Frederick, Maryland 21701-5012		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO 63807A		PROJECT NO. 344- 63807D993✓	
		TASK NO CA		WORK UNIT ACCESSION NO 143	
11. TITLE (Include Security Classification) THE PERSONAL MONITOR AND COMMUNICATOR (PMC)					
12. PERSONAL AUTHOR(S) W. A. Tacker, Jr., M.D., Ph.D.					
13a. TYPE OF REPORT Final Report		13b. TIME COVERED FROM 8/1/87 TO 10/31/87		14. DATE OF REPORT (Year, Month, Day) 1989 December 31	
15. PAGE COUNT					
16. SUPPLEMENTARY NOTATION Subtitle: An Electronic Dogtag					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	RA V; Artificial Intelligence; Expert Systems; Field Medical Dev; Med Monitor; Vital Signs; Computers		
23	05				
09	01				
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This report describes the development of a life detector system for use in the conventional or chemical battlefield. The system can detect motion, and heart rate in military casualties either in conventional or chemical warfare clothing and gear, without violation of the soldier's chemical protective barrier. The system includes two components both of which were delivered to the Army for concept evaluation: the Personal Monitor and Communicator (PMC) which is worn on the wrist of each soldier and the Hand Held Monitor (HHM) which is carried by the medic. Plethysmographic signals from stainless steel electrodes in the wristband of each PMC are used to determine the heart rate of the soldier and/or whether the soldier is moving. Radio communication links allow the HHM to display the data to medics at a range of 5 feet. The system is operational under conditions of conventional or chemical battlefield warfare at all levels up to an including MOPP-4. Also developed to prototype (breadboard) stage was a third piece of hardware called a multimonitor, which is designed to simultaneously monitor multiple casualties wearing PMCs at a field medical care station.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input checked="" type="checkbox"/> DPC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Mrs. Virginia M. Miller			22b. TELEPHONE (Include Area Code) (301) 663-7325		22c. OFFICE SYMBOL SGRD-RMT-S

FOREWORD

Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the U.S. Army.

N.A. Where copyrighted material is quoted, permission has been obtained to use such material.

N.A. Where material from documents designated for limited distribution is quoted, permission has been obtained to use the material.

☒ Citations of commercial organizations and trade names in this report do not constitute an official Department of the Army endorsement or approval of the products or services of these organizations.

N.A. In conducting research using animals, the Investigator(s) adhered to the "Guide for the Care and Use of Laboratory Animals," prepared by the Committee on Care and Use of Laboratory Animals of the Institute of Laboratory Animal Resources, National Research Council (NIB Publication No. 86-23, Revised 1985).

☒ For the protection of human subjects, the Investigator(s) have adhered to policies of applicable Federal Law 45 CFR 46.

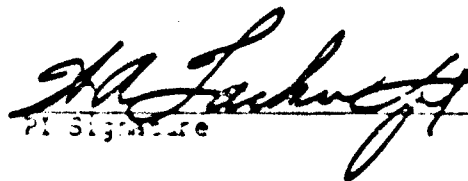
 1-8-80
PI Signature Date

TABLE OF CONTENTS

	PAGE
Introduction	5
Nature of Problem	6
Background of Previous Work	7
Remote Triage System	7
Purpose of Present Work	18
Description of Deliverable Items	18
Revised Scope of Work in 1988	19
Methods of Approach	20
Theory	20
Results Obtained	24
Personal Monitor/Communicator	24
Hand Held Monitor	24
Multimonitor	32
Conclusions	38
References	38
Technical Data Package	Attached
Software for Multimonitor	Attached

Accession For	
NTIS CRAS	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution	
Availability Codes	
Dist	Avail and/or Special
A-1	



Introduction

This final report is organized into two parts. The first part is an executive summary which is contained in the first 37 pages of the report, and the second part is the complete and full Technical Data Package provided to the U.S. Army according to the technical standards required for reporting this completed contract effort. The executive summary is included because the complete manuscript (Technical Data Package) is so long as to be very time consuming to read in its entirety, and the abstract is too short for even an overview of such an extensive effort.

Nature of the Problem

Medical care of casualties on the battlefield is a major challenge to military operations under all conditions, but is especially difficult with the risk of chemical agent exposure. In particular the currently deployed chemical protective ensemble to be worn by U.S. Army personnel in a chemical agent environment has created a unique problem. When fully protected (that is at MOPP-4 level) by the complete ensemble, it is not possible for the medic to determine whether a motionless casualty is alive or dead. Neither taking a pulse, nor seeing chest motion with breathing is possible due to the bulky, thick material of the chemical defense suit. The mask prevents the medic from detecting air flow during respiration.

Of course, the chemical defense gear cannot be removed from the casualty for the purpose of gaining access to the anatomical sites suitable for detecting the pulse or respiration because removal of the gear without decontamination would expose the casualty to the chemical agent. Consequently, there is a need for a method to determine whether or not a casualty in full chemical protective gear is dead or alive without violating the integrity of the protective gear. This is the problem addressed in the project DAMD 17-87-C-7195.

Background of Previous Work

The Hillenbrand Biomedical Engineering Center of Purdue University has developed a new method of detecting life signs using a sensor system worn on the wrist. Several events can be detected, including motion, heart rate and respiration using this sensor. We proposed in 1985 to develop a prototype system for field testing which would provide remote casualty triage and location using a 2-way radio link, the new sensors, and a stimulus-response sequence whereby the soldier could indicate his or her status to field medical personnel.

Remote Triage System

We proposed and showed feasibility for remote triage of casualties in the battlefield. We also proposed further development to achieve a practical and fieldable system. The system envisioned would include sensors worn on the wrist of each soldier (Figure 1) to determine responsiveness, motion, heart rate, and respiratory rate. A computer based expert system at a central station (Figure 2) uses these data to triage soldiers into categories of 1) functional, 2) wounded but stable, 3) needing immediate evacuation, 4) alive, but fatally wounded, and 5) dead. Display of this information is provided in tabular or graphic form on a terminal at central station. The system operates under control of the central station operator who uses the menu driven program in the central computer. A major capability of the system is to determine consciousness and responsiveness remotely. Also, the 2-way radio link allows the location of the casualties using a graphic display of the battlefield.

For example after friendly forces have achieved control of the battlefield, the central station interrogates all of the soldier PMC's by radio transmission (Figure 3). A non-painful, safe, electrical stimulus is applied to the soldiers' wrists from electrodes A and B on the inner surface of the wristband (Figure 3). If the soldier is all right, he pushes the "OK" button. If the OK button is not pushed (Figure 4), impedance sensors switch on in the wristband and record motion, heart rate and respiration. All data is processed and stored in the wristband's microprocessor.

A second radio signal is then sent to all wrist units which instructs the wrist units to serially transmit back the data in a short data bit sequence (Figure 5). The central station expert system then categorizes all soldiers, and determines location of the casualties only. Non-casualties could be locked out of the locator capability.

Data can be presented at central station in the form of detailed tabular data (Figure 6), or bar graph summaries (Figure 7), as well as display of casualty location (Figure 8). Thus medics, ambulances and other resources can be deployed to best advantage, favorably affecting mortality and morbidity of casualties, and decreasing risk to medical personnel.

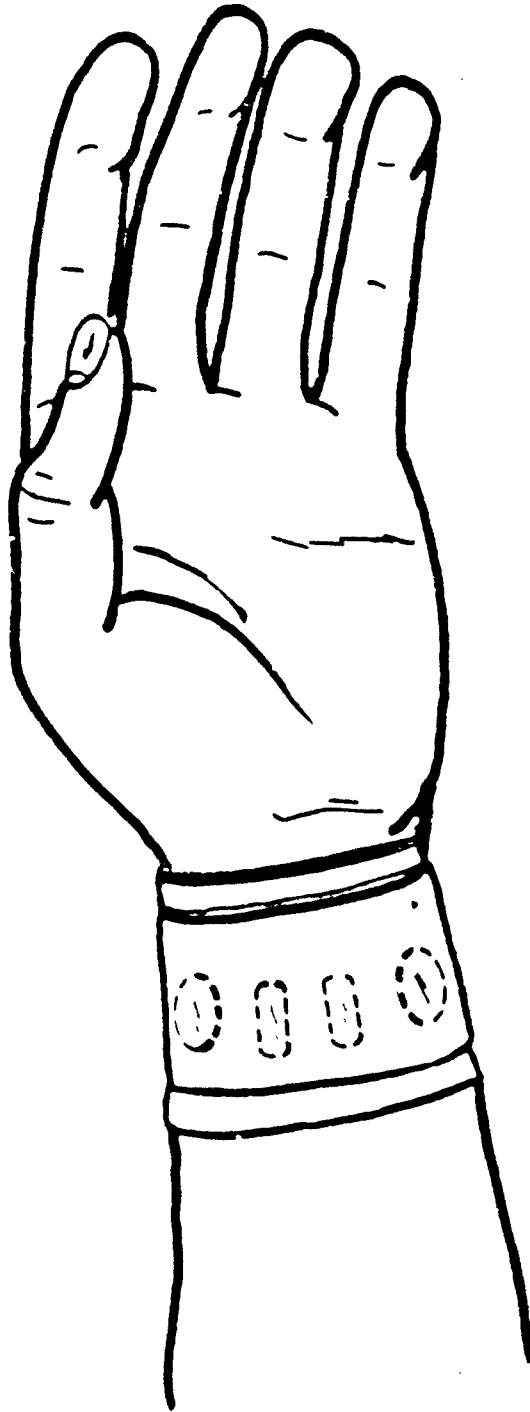
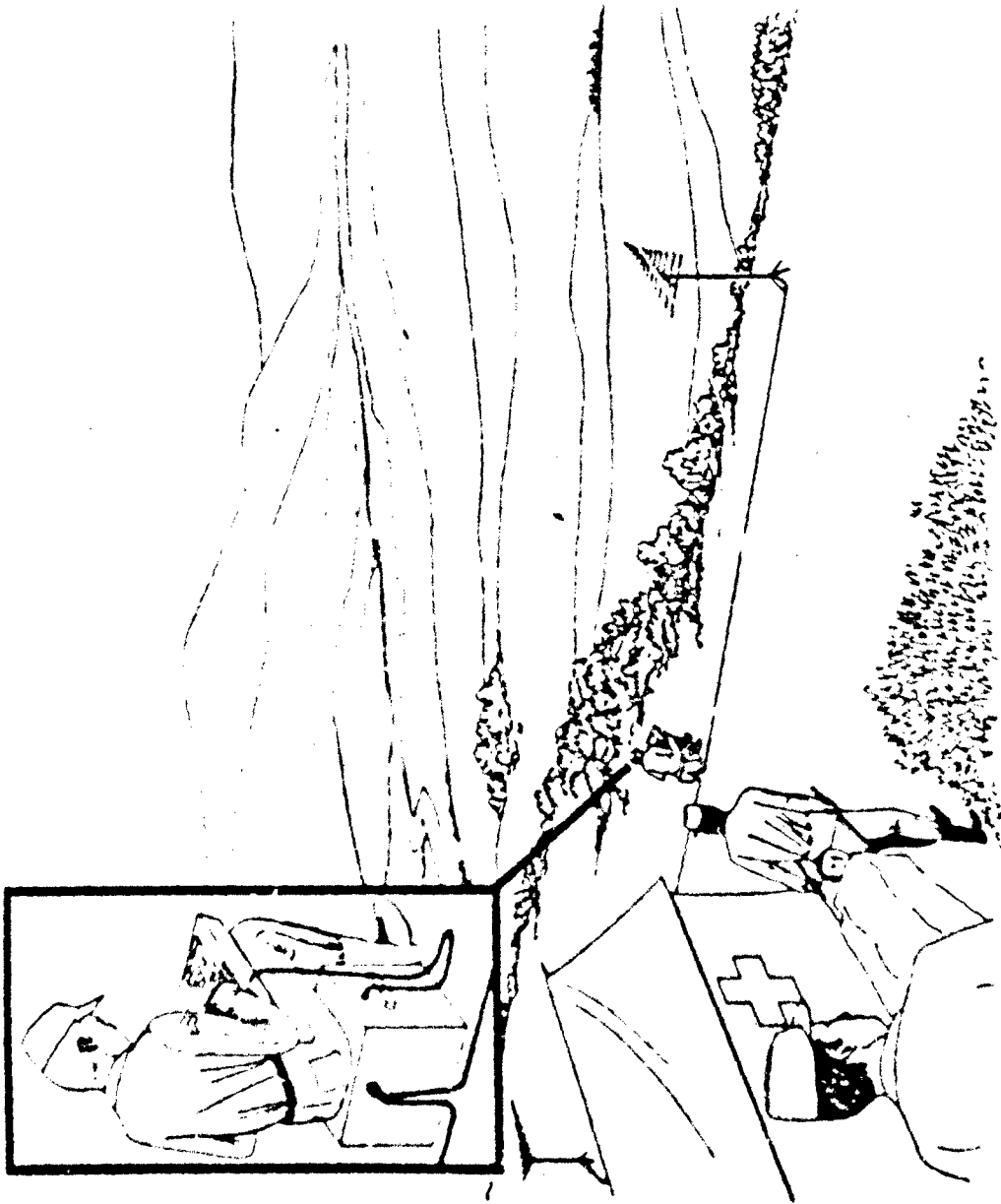


Figure 1. Wristband diagram to show electrode/sensor locations.



Central station for remote triage system showing radio link antennas, and central display and control unit operated by a medic.

Figure 2.

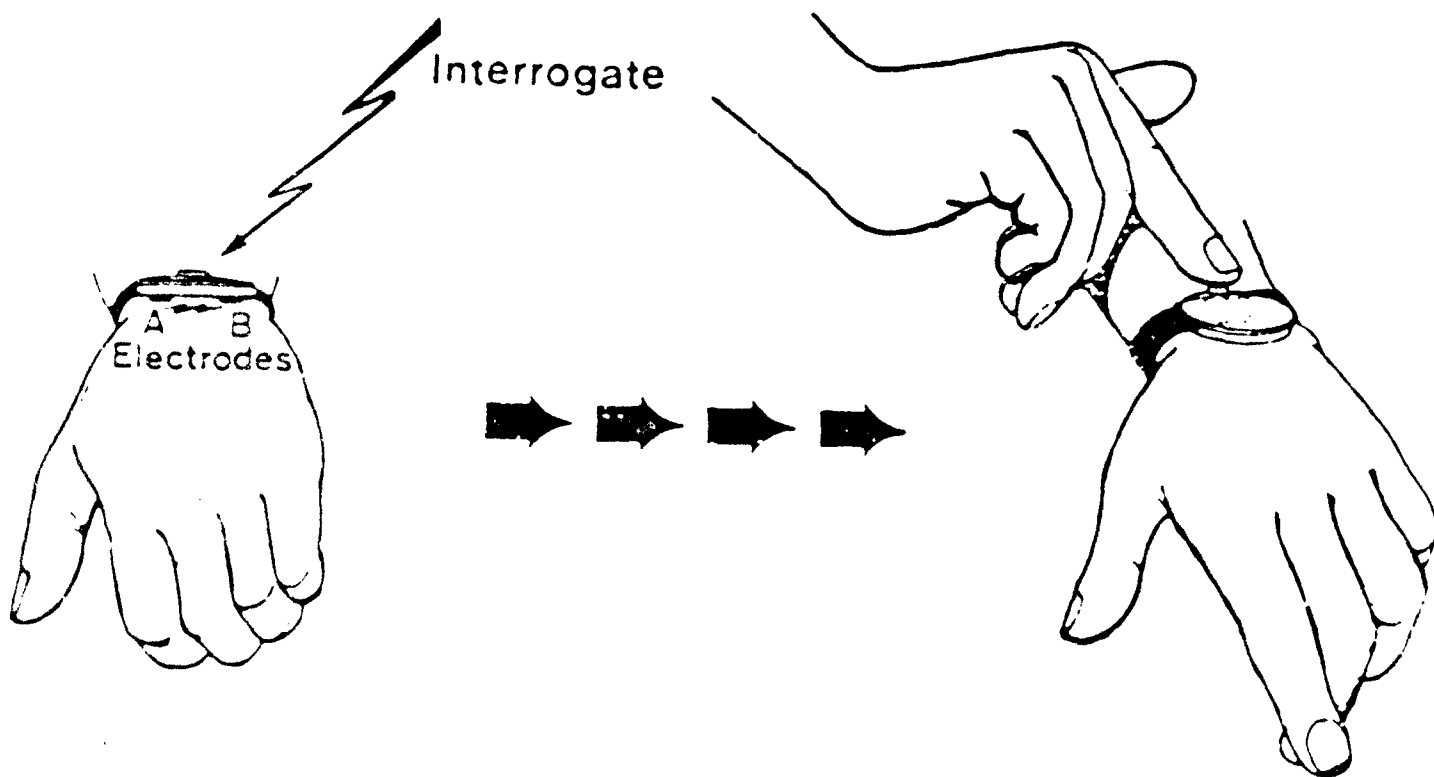
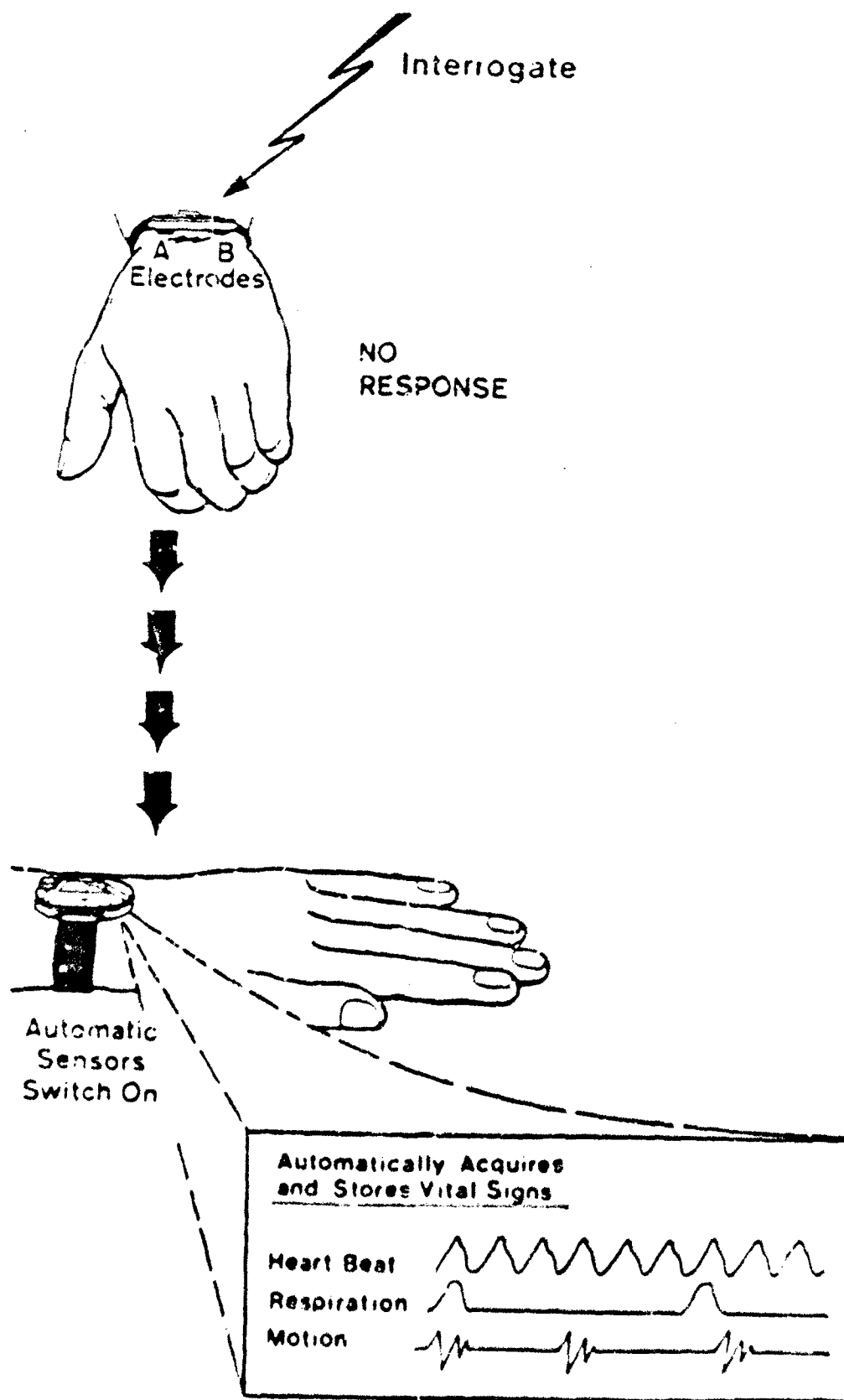


Figure 3.

The wrist unit is activated by a "interrogate" radio signal (on left), and if soldier feels stimulus, the "OK" button is pushed.

Figure 4.

IF the "OK" button is not pushed by the soldier, the sensors switch on, obtain physiologic data, and store the data in memory.



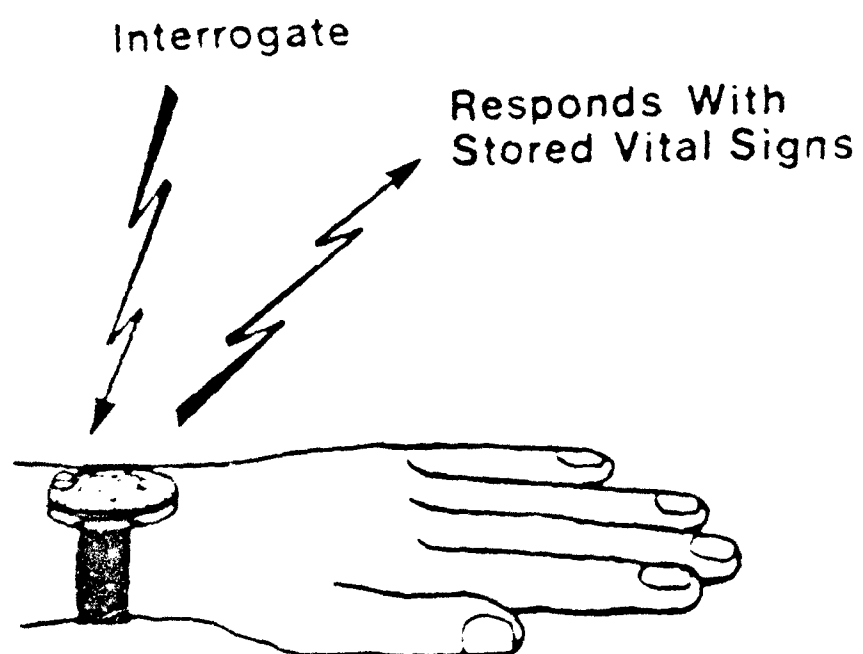


Figure 5.

Second "Interrogation" signal activates sequential data transmission from wrist unit to central station.

Information Display

PMC Interrogation System Condition: Detailed Examination of Wounded That Need Evacuation (6)				
ID #	Responded	Moving	Heart Rate	Respiration Rate
014	no	yes	190	??
045	no	yes	160	30
061	no	no	055	24
106	no	no	170	30
171	no	no	055	10
204	no	no	120	08

Figure 6. Tabular presentation of casualty data at central station display.

Information Display

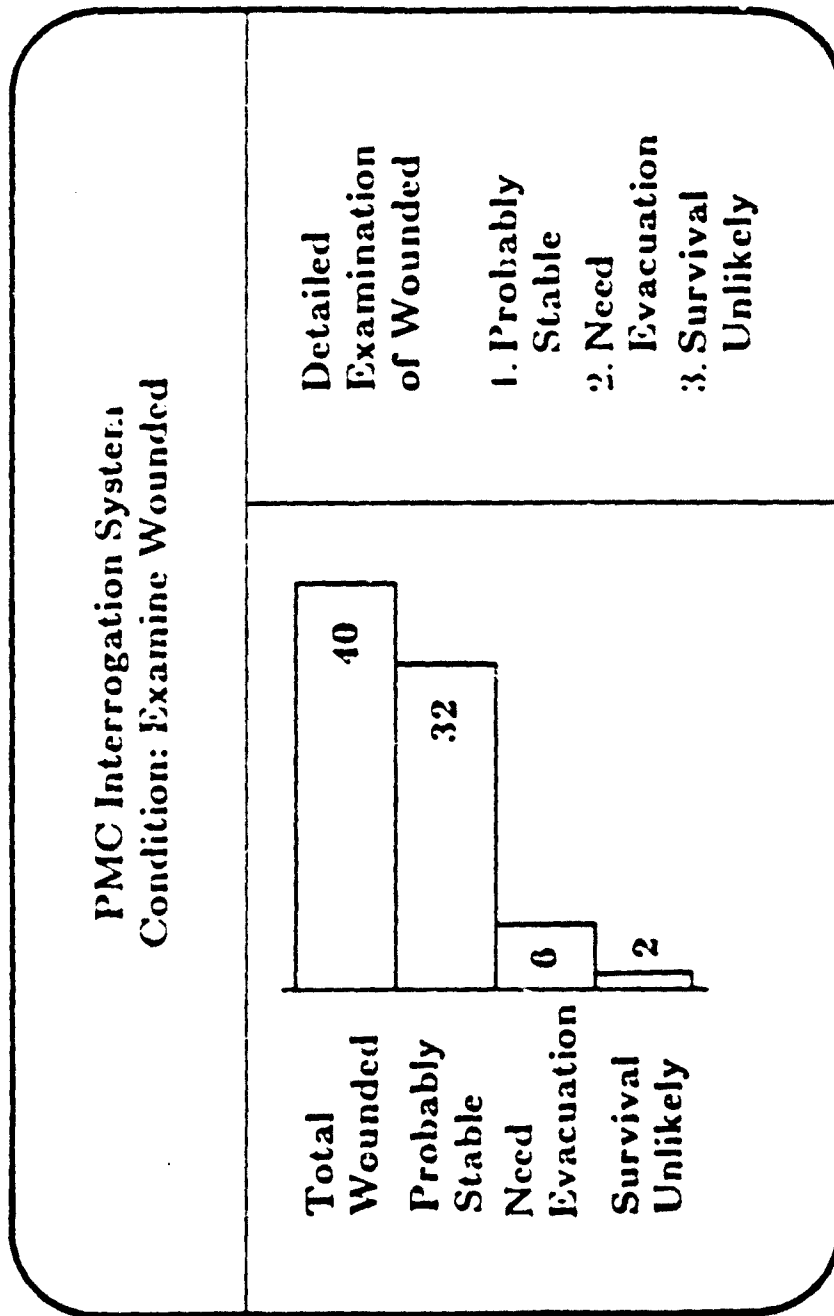


Figure 7. Graphic presentation of casualty data at central station display.

Information Display

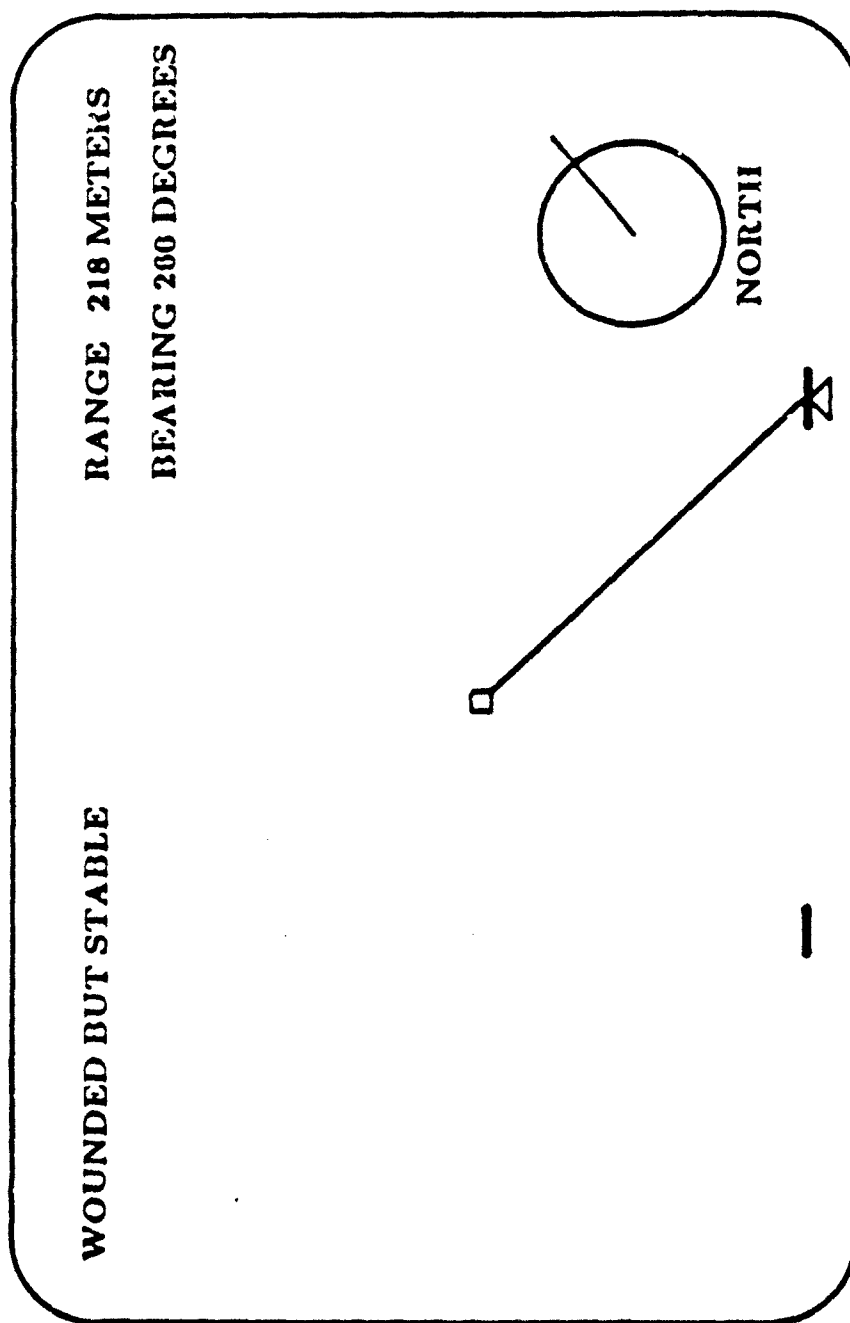


Figure 8. Display of casualty location with range and bearing from central station, as well as triage status.

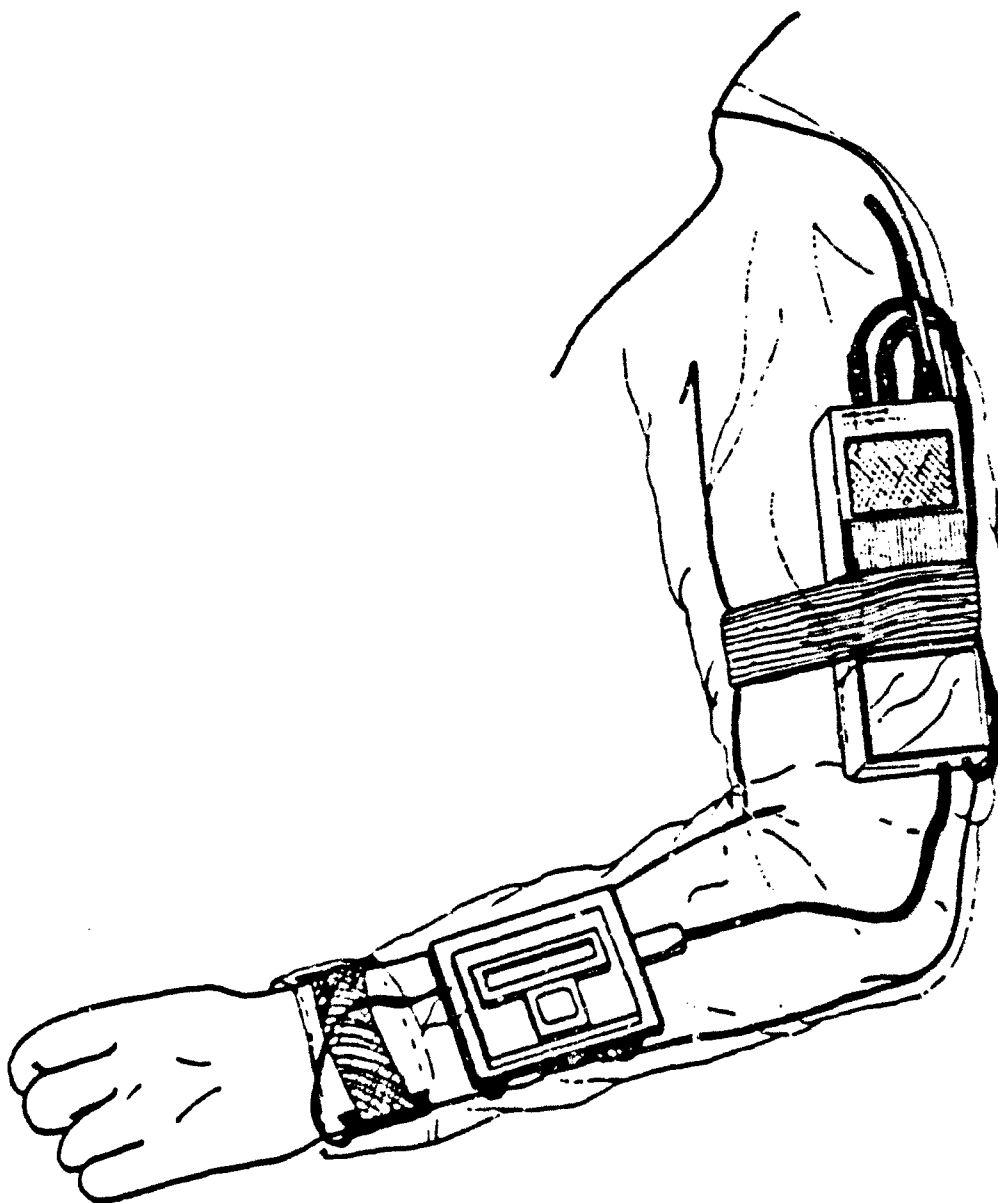
In summary, the individual soldier's equipment is worn on the wrist, and includes a 2-way radio link, a stimulator which applies a safe, non-painful electrical stimulus to the wrist, a response switch, a tetrapolar, dry-electrode, impedance measuring plethysmograph, and a microprocessor-based signal processor which determines subject motion, heart rate and respiratory rate. The central station consists of 2 antennas and 2-way radio links, menu-driven software for the expert system, and tabular or graphic displays on a small portable computer. Graphic display includes the soldiers' triage category and location.

The system is envisioned to be optimized for operation after a battle, or during lulls in action, when deployment of medical personnel is most practical, and security of the information can be best assured. Finally, the system will be operational under conditions of conventional or chemical battlefield warfare at all levels up to and including MOPP-4.

On February 26, 1986 a prototype system was demonstrated to the Army in W. Lafayette, Indiana, on the Purdue Campus. The prototype, soldier-worn hardware as demonstrated, is shown in Figure 9.

Figure 9.

Prototype hardware demonstrated during earlier work on field medical care of soldiers.



Purpose of Present Work

Because of the need for a system to detect life in a casualty wearing chemical defense equipment our development effort was redirected away from the remote triage system at the request of the Army, and funded in 1986 to provide a life detector system with these characteristics for concept evaluation:

1. A complete functional system comprised of 2 pieces of hardware, the wristband (PMC) and the hand-held interrogator, or monitor (HHM).
2. Two (2) systems to be provided, i.e. 2 wrist bands and 2 interrogators.
3. The system will be operable with both the wearer and medic in NBC protective clothing ensemble, or with the wearer in a chemical warfare agent protective wrap and the medic in NBC protective clothing ensemble. No penetration, or removal, of the chemical protective gear will be required for use.
4. Both hardware items will be battery operated, with replaceable batteries.
5. The interrogator will provide readout of heart rate on an LCD.
6. The system will be operable at a range of up to 5 feet between the interrogator (medic) and wearer (casualty).
7. No direct contact will be required between the user (medic) and wearer (casualty).
8. The system will be designed to discriminate as much as possible between the heart rate and expected artifacts of noise, vibration, motion, and environmental electronic signals.

Description of Deliverable Items

Brass-board hardware will be provided including two (2) PMC units and two (2) hand held transmitter/receivers suitable for use by a medic or other operator. The PMC's are to be worn on the wrist of a subject, and will determine heart rate from the subject and transmit the heart rate to the hand-held receiver. The system will be designed to operate even when the subject is at MOPP-4 level and/or in a chemical protective wrap. The range of operation will be at least 5 feet. The system will be capable of determining which of the 2 PMC's is being received when both units are in close proximity. In addition, instructions for use of the system will be provided.

The size of the wrist-worn and medic operated components to be delivered for concept testing must be reduced, compared to the triage prototype hardware demonstrated in 1986, but this hardware will not be as small as the system could be made for widespread deployment. (These units can be made 40 to 60% smaller than the concept evaluation units, because custom chips and hybrid circuits can be incorporated after design is finalized for large scale production.)

Revised Scope of Work in 1988

It is important to note that a substantial revision in scope of work was requested and negotiated by the Army, and agreed to on May 6, 1988. This revised scope specified an increased number of deliverable items including eight more wrist units, one more hand held unit and one prototype multimonitor. Also, an extension of time for delivery was given, inasmuch as the Army tests were not scheduled until 1990. Also, there was a breadboard "multimonitor" originally specified in the contract for a "demonstration only" at Purdue, and in the revised scope of work this breadboard was specified as a deliverable item. However, the multimonitor was specifically excluded from the Technical Data Package. The envisioned multimonitor use is at a battalion aid station, or other suitable area, and it is designed to monitor multiple PMCs simultaneously, and automatically.

Methods of Approach

The technology used for this life detector system was very similar to that described previously for the remote triage PMC systems, using tetrapolar impedance sensors on the wrist. The electrodes are of dry stainless steel. The complex impedance signal from the wearer's wrist is processed in the PMC unit for heart rate determination. The state-of-the-art Motorola MC68HL11 CMOS chip is the heart of the wrist-worn part of the system. Radio-frequency communication links used are adequate for the short-range requirements.

Theory

The heart rate is measured by monitoring the peripheral pulse from the wrist using a plethysmograph. Four electrodes (referred to as a tetrapolar electrode system) are placed around the wrist. As the circulatory system pumps blood through the arteries at the wrist, the impedance changes at the electrodes can be sensed by electronic circuitry. These changes are interpreted as pulses. This method is easy to apply, is non-invasive and is only slightly affected by variations in temperature and barometric pressure. Figure 10 shows a typical complex plethysmographic signal. Signal processing separates the cardiac and respiratory signals as shown in Figure 11. Heart rate can then be counted. Figure 12 shows the effects of motion to produce large signals which document motion of the subject.

The wristband of the device, called a PMC, contains the four electrodes. The device functions properly only when all four electrodes are in full contact with the skin. The wrist impedance signal measured by the plethysmograph is connected to an A/D converter contained in the microprocessor. The processor accesses a software algorithm to analyze the digital signal and compute the heart rate. The resulting value is stored in memory. This real time process continuously updates the heart rate value while the PMC unit is activated.

The heart rate is displayed on the LCD of the HHM when a transmission link exists between the HHM and the PMC. The transmission link is established when the HHM is placed within 2 inches of a selected PMC and the *ACTIVATE* button of the HHM is pressed. This action causes a coil in the HHM to induce a current in a coil in the PMC which in turn activates the PMC. The PMC then begins the data collection and the heart rate calculation. The heart rate is transmitted via a radio frequency transceiver to the HHM and displayed on the LCD. The HHM continues to request information from the selected PMC and the PMC continues to transmit updated heart rate information until the HHM-PMC link is broken. A break occurs when the HHM activates a new PMC or when the HHM is turned off. All data collection and heart rate calculations occur automatically. No outside intervention is necessary.

Wrist Impedance Monitor

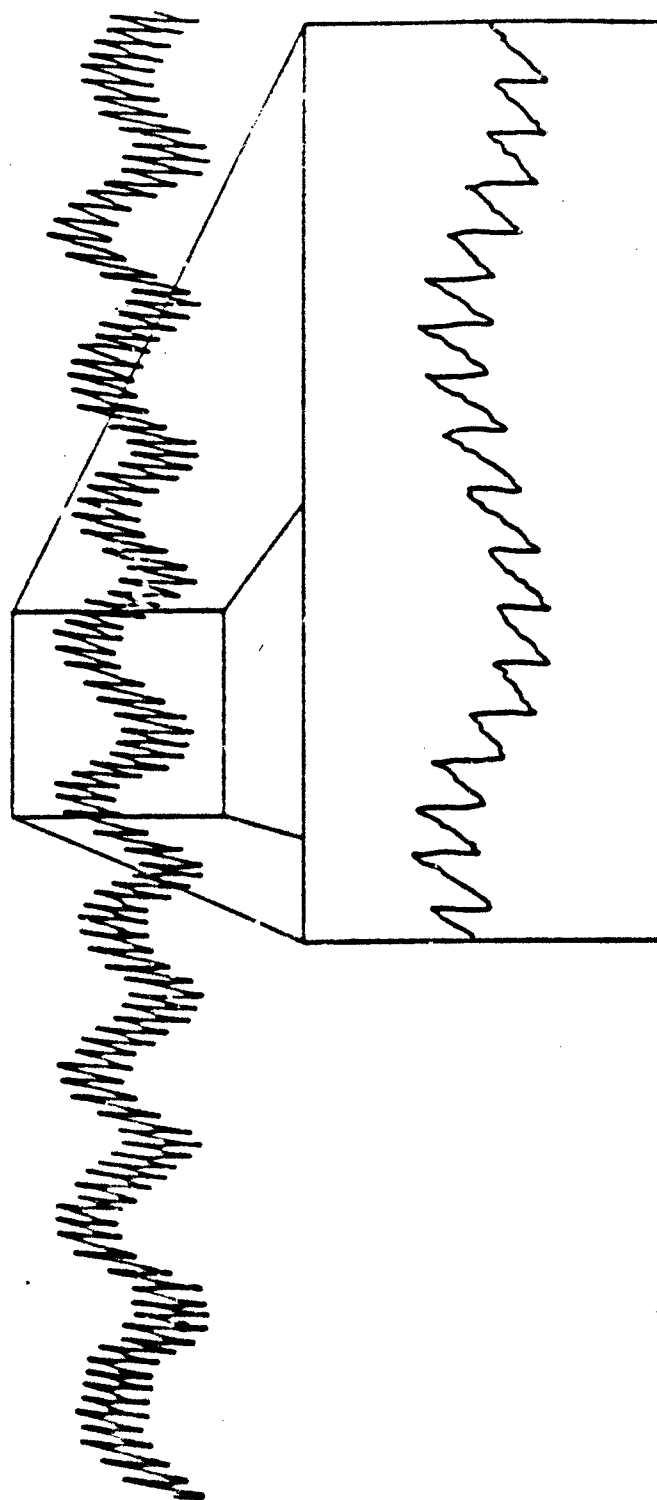


Figure 10. Typical signal from a motionless subject.

Figure 11.

Separation of complex (upper) signal into cardiac (middle) and respiratory (lower) components.

Wrist Impedance

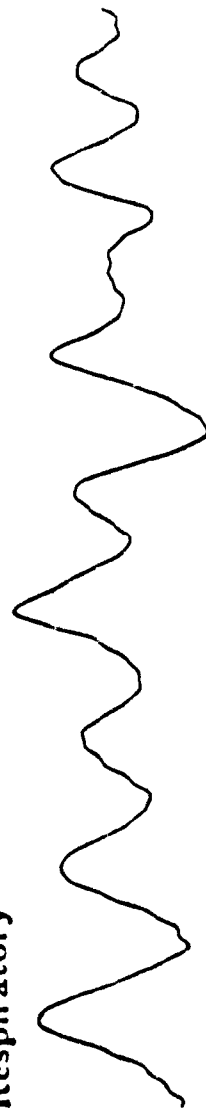
a) Composite



b) Cardiac



c) Respiratory



1 Minute

Wrist Impedance

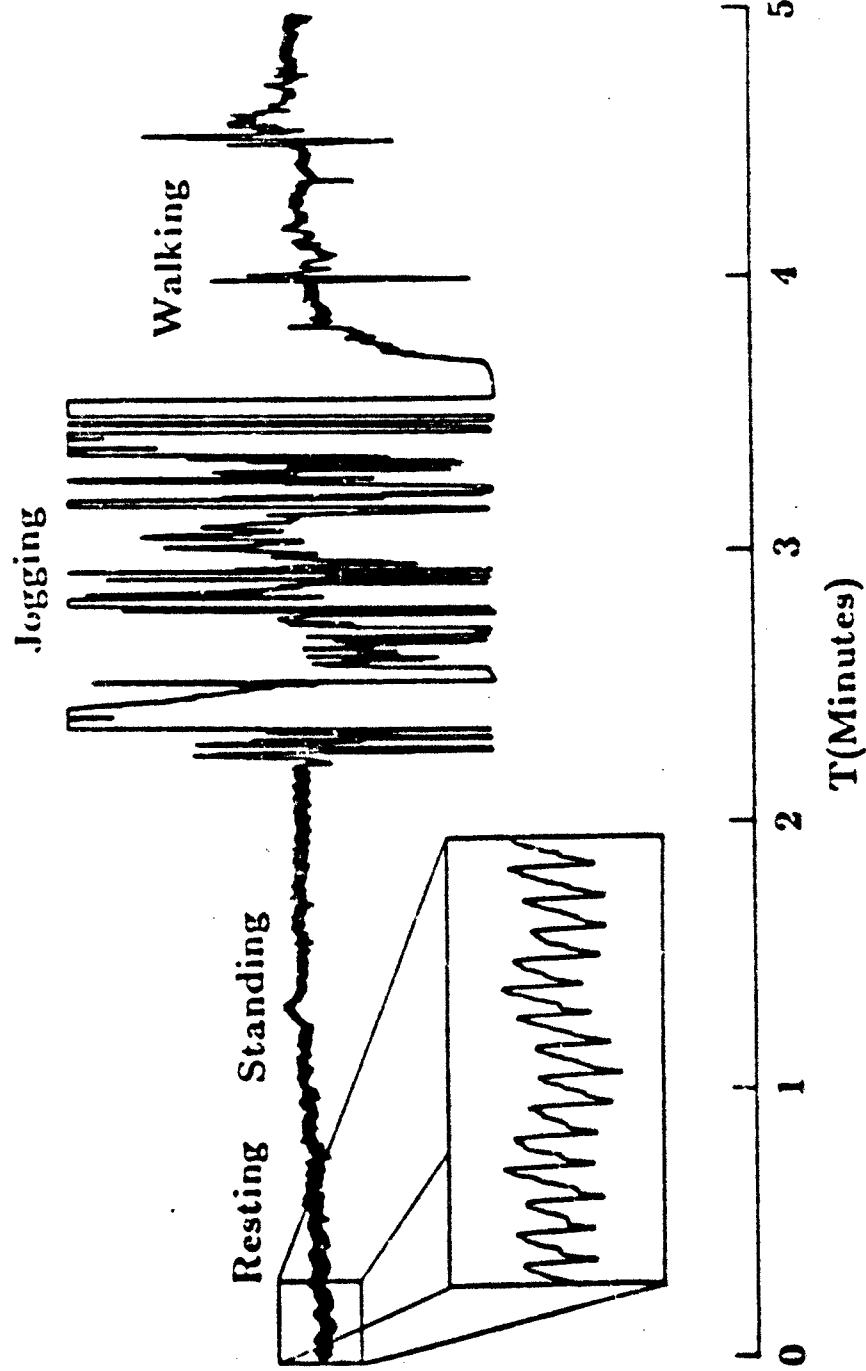


Figure 12. Example, in center of figure, of large amplitude motion signals.

Results Obtained

As requested, the system was developed, demonstrated at an acceptance meeting on October 31, 1989 at Purdue University, and delivered to Ft. Detrick, MD. in November, 1989.

Personal Monitor/Communicator

The PMC is composed of a rectangular box connected by wire leads to a wristband. The wristband contains the tetrapolar dry-electrode impedance measuring plethysmograph necessary for monitoring the soldier. The band wraps around the wrist of the soldier and must be in direct contact with the skin. The box contains a 2-way radio link and a microprocessor. The unit is powered by batteries which fit in a compartment on the side of the box. The box needs to be affixed to the forearm (left or right) of the soldier. There is no need for the box to be in direct contact with the skin. The entire PMC assembly can be worn under a chemical defense jacket or other clothing. See Figure 13.

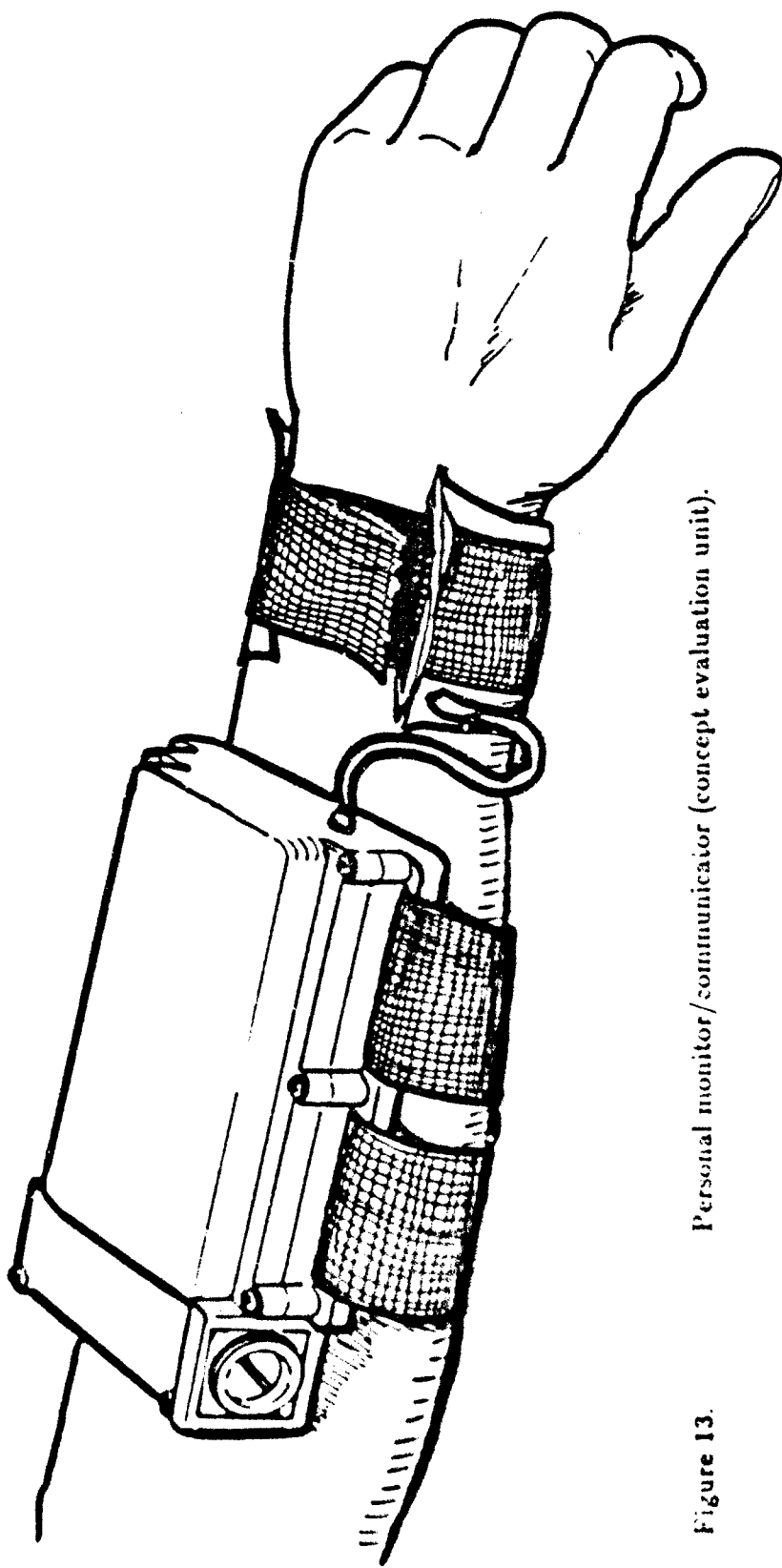
Hand Held Monitor

The HHM consists of a single rectangular box with a removable antenna (Figure 14). Located on the front face of the HHM are two switches (*ON-OFF* switch and *ACTIVATE* button) and a display panel. The *ON-OFF* switch controls the power to the HHM. The *ACTIVATE* button causes the HHM to gather information from a specified PMC. The rectangular liquid crystal display (LCD) panel displays the status of the soldier being monitored and the status of the PMC. An additional switch is located on the side of the HHM nearest the *ON-OFF* switch. This switch turns on the background light for night time use of the LCD. Two battery compartments are located on the HHM. The battery compartment located on the top of the unit near the antenna provides power for background lighting of the liquid crystal display. The battery compartment on the bottom of the HHM is the main power supply. The bottom battery compartment cover can be seen in Figure 14.

The HHM indicates it is on by displaying the following message:

Hand Held
Monitor (HHM)

This message remains on the screen for only a few seconds.



Personal monitor/communicator (concept evaluation unit).

Figure 13.

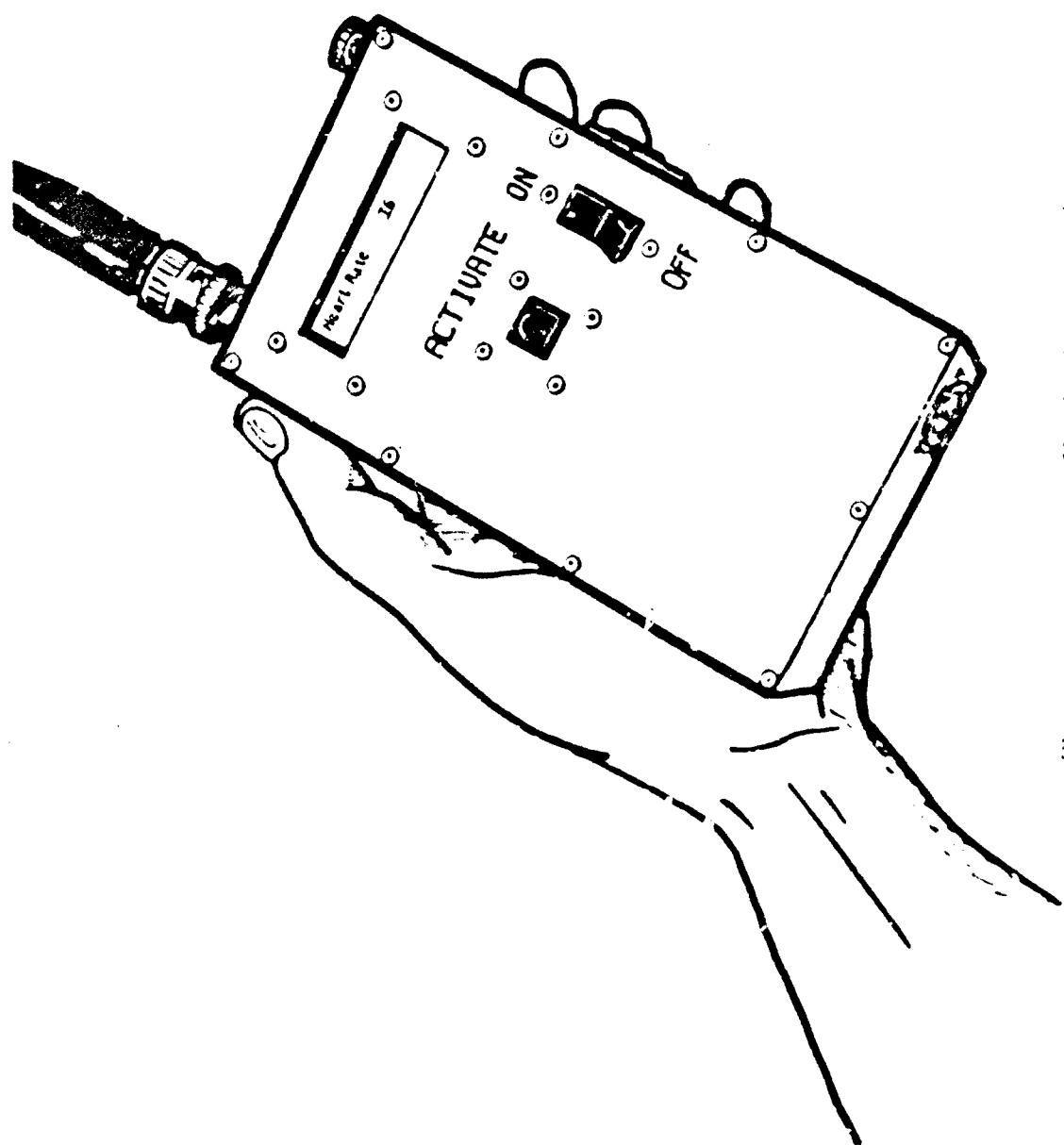


Figure 14. Hand held monitor (concept evaluation unit).

The HHM then displays the following instructions:

Put HHM Near PMC
Press *ACTIVATE*

The back side of the HHM is held within 2 inches of the rectangular box of the PMC that is to be monitored, as shown in Figure 15. With the HHM in this location the operator briefly presses and releases the *ACTIVATE* button on the front of the HHM. The HHM then displays the message:

Attempting
PMC Activation

When the activate procedure works, the HHM displays the message:

PMC ACTIVATED

The HHM can now be moved a few feet away from the PMC.

The HHM will then attempt to communicate with the designated PMC and will display the message:

Attempting PMC
Communication

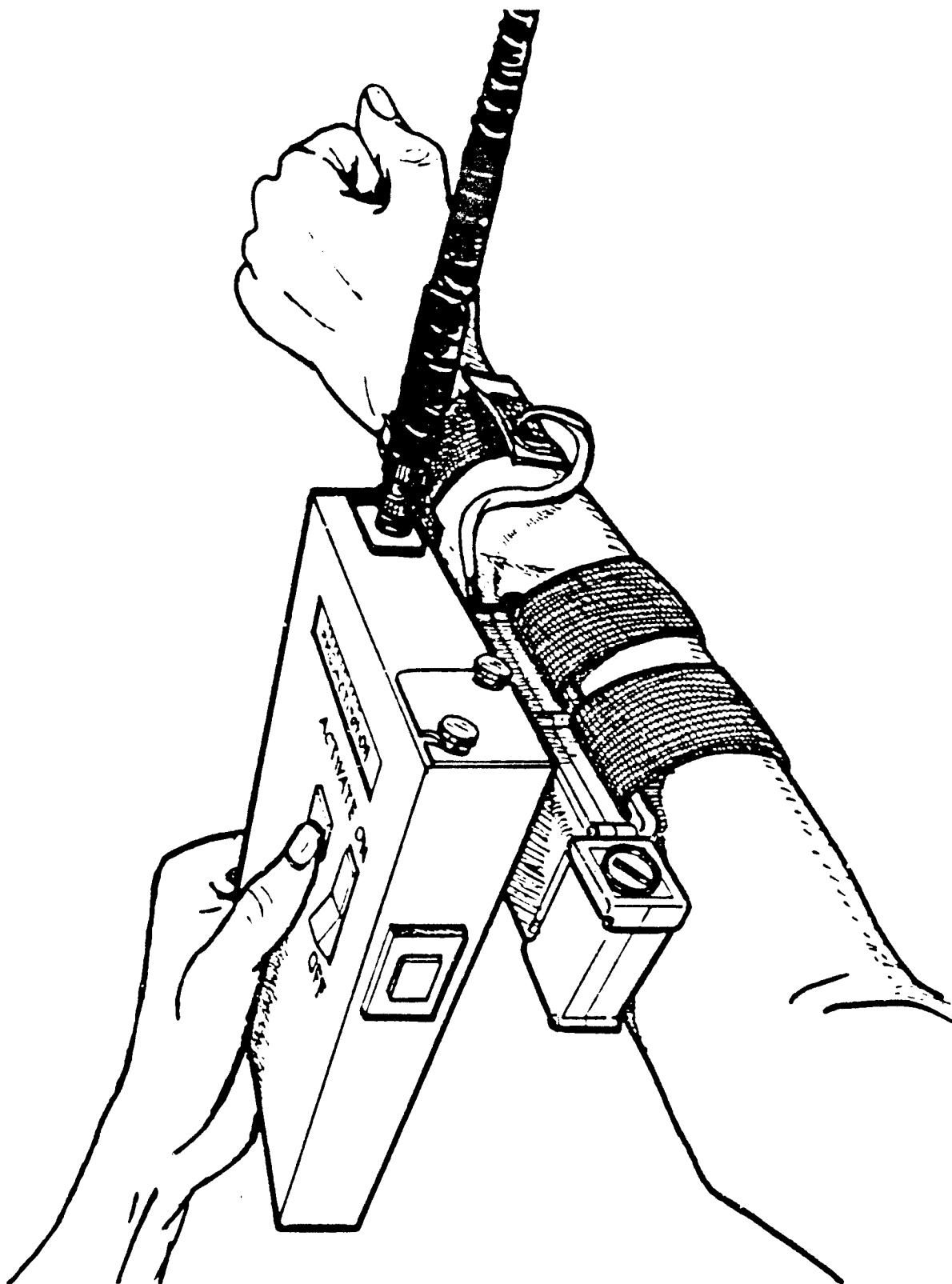


Figure 15. Position of PMC and HIM for system activation. In this figure the chemical defense jacket is not shown. However, the system will transmit through the jacket. (See Figure 16).

When communication is established between the HHM and the PMC, the HHM begins to display messages concerning the soldier's status. If the sensed heart rate is reasonable, then the message:

Heart Rate #

where the character '#' is replaced by the soldier's heart rate, appears. See Figure 16.

Other messages that may occur during this phase of monitoring include:

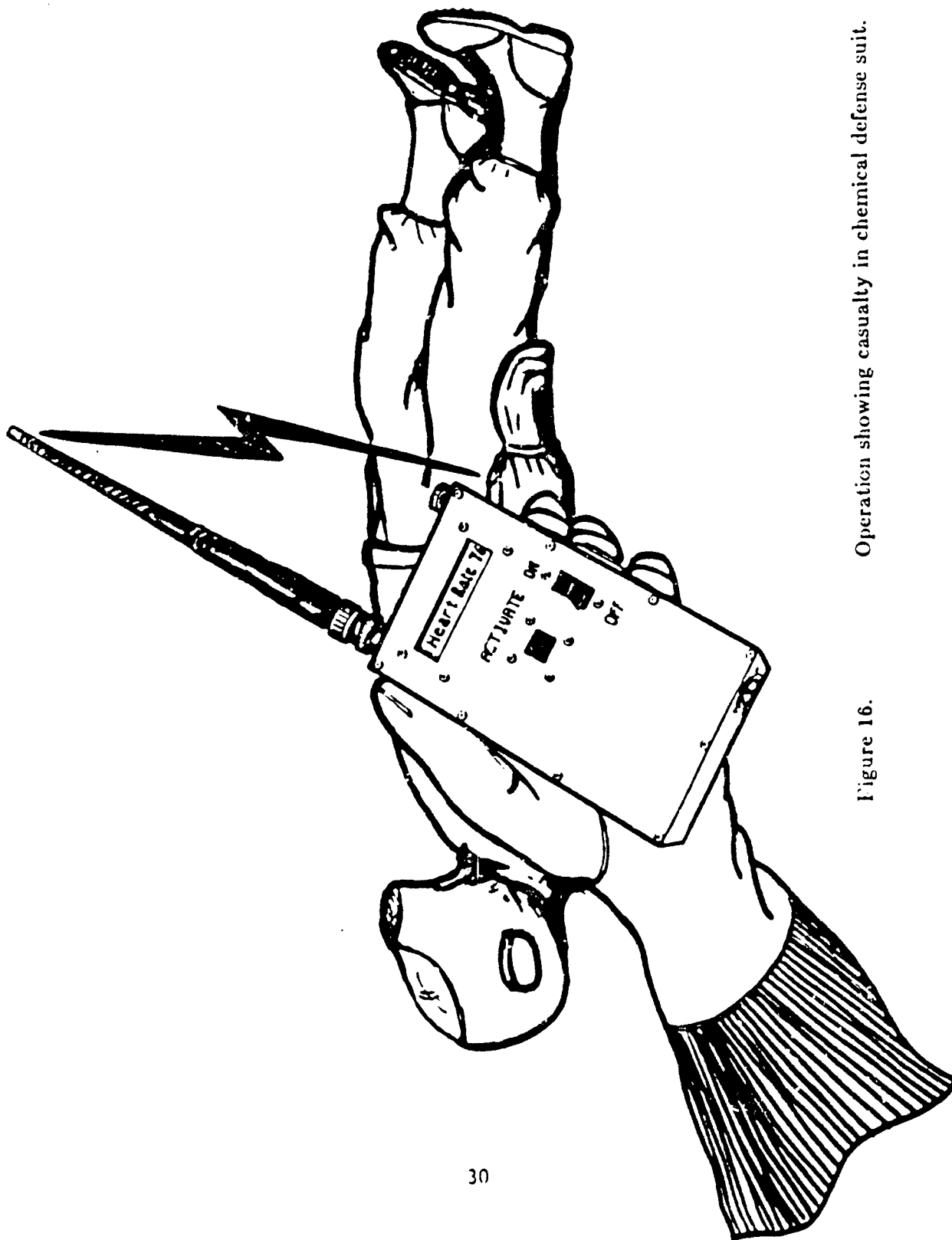
Low Pulse

or

Motion

Messages are displayed approximately every 15 seconds after the PMC has been activated. If motion is present, the heart rate may not be displayed.

Monitoring a new PMC may begin during any point of the HHM operation. To monitor a new PMC, the back side of the HHM is held approximately 2 inches above the rectangular box of the new PMC and the *ACTIVATE* button is pressed. The HHM now ignores the previously active PMC and monitors the newly selected PMC. More than one HHM may monitor a single PMC.



Operation showing casualty in chemical defense suit.

Figure 16.

Conditions may require the HHM software to make several attempts to activate the selected PMC. The retransmission message:

Attempting
Retransmission

will appear on the screen until the procedure fails or the PMC is activated.

If the activate procedure fails, then the following message is displayed:

ERROR--Repeat
Activate Steps

and the activate procedure must be repeated.

After the PMC is activated, several attempts may be made by the HHM to communicate with the PMC. During these attempts, the following message will appear on the LCD screen:

Attempting
Retransmission

This message indicates the HHM has not successfully prompted the activated PMC.

If the HHM is unable to establish communication with the PMC then the HHM displays the message:

Lost
Communication

After this message, the PMC must be re-activated so the HHM displays the message:

Put HHM Near PMC
Press ACTIVATE

If the message:

Check
Wrist Strap

appears, then the PMC is not properly strapped to the soldier. The HHM will not be able to determine the soldier's status when this problem exists.

If the message:

Low PMC Battery

appears, then the batteries in the PMC unit are weak and must be replaced.

Multimonitor

The front view of the prototype monitor is shown in Figure 17. This device can be used to monitor the status of several soldiers in the immediate area, and for example, could be used in the battalion aid station to allow one medic to monitor multiple casualties, even under conditions of chemical warfare. It is called the Multi-Monitor. The device will automatically and periodically check the heart rate and motion status of all soldiers being monitored. The Multi-Monitor is a battery-powered portable device with integral radio, keypad, display and audible alarm.

When the multimonitor is turned on by pressing the off-on rocker switch (see Figure 17) the instructions for use appear on the display as shown.

To activate a PMC, the multimonitor is held over the wrist unit (within 2 inches) and the ADD button is pushed. After about 20 seconds the casualty's identification number, heart rate, and motion status will be displayed on the screen.

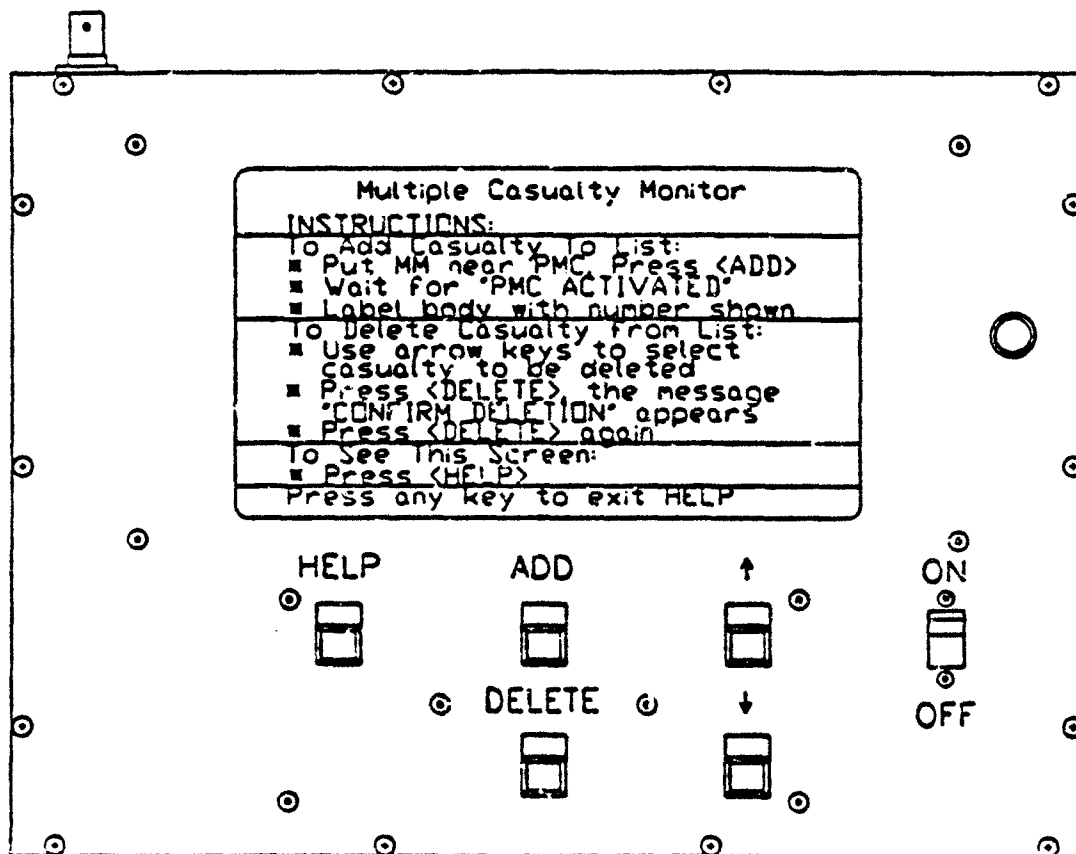


Figure 17.

Multiple Casualty Monitor front view.

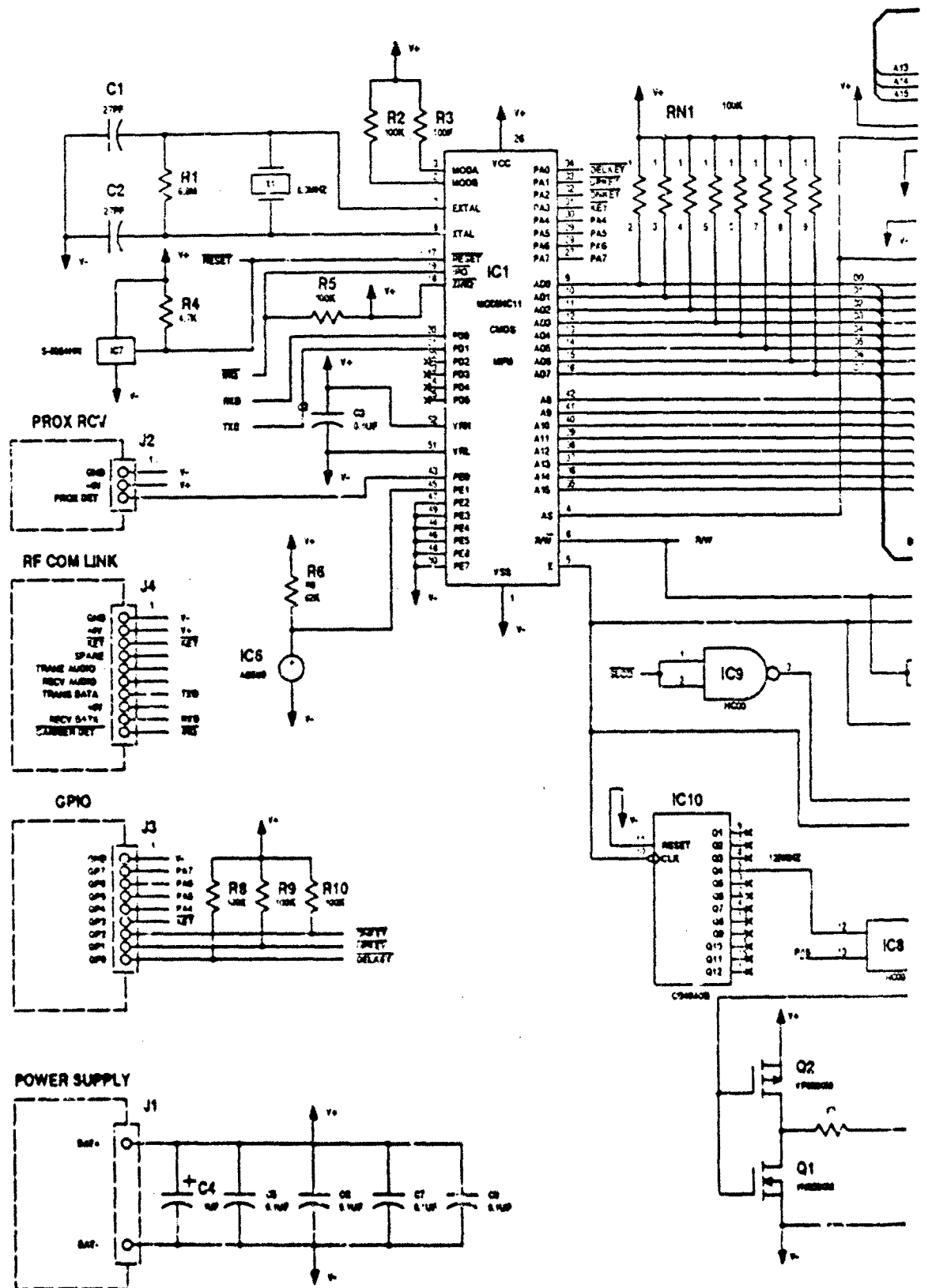
1

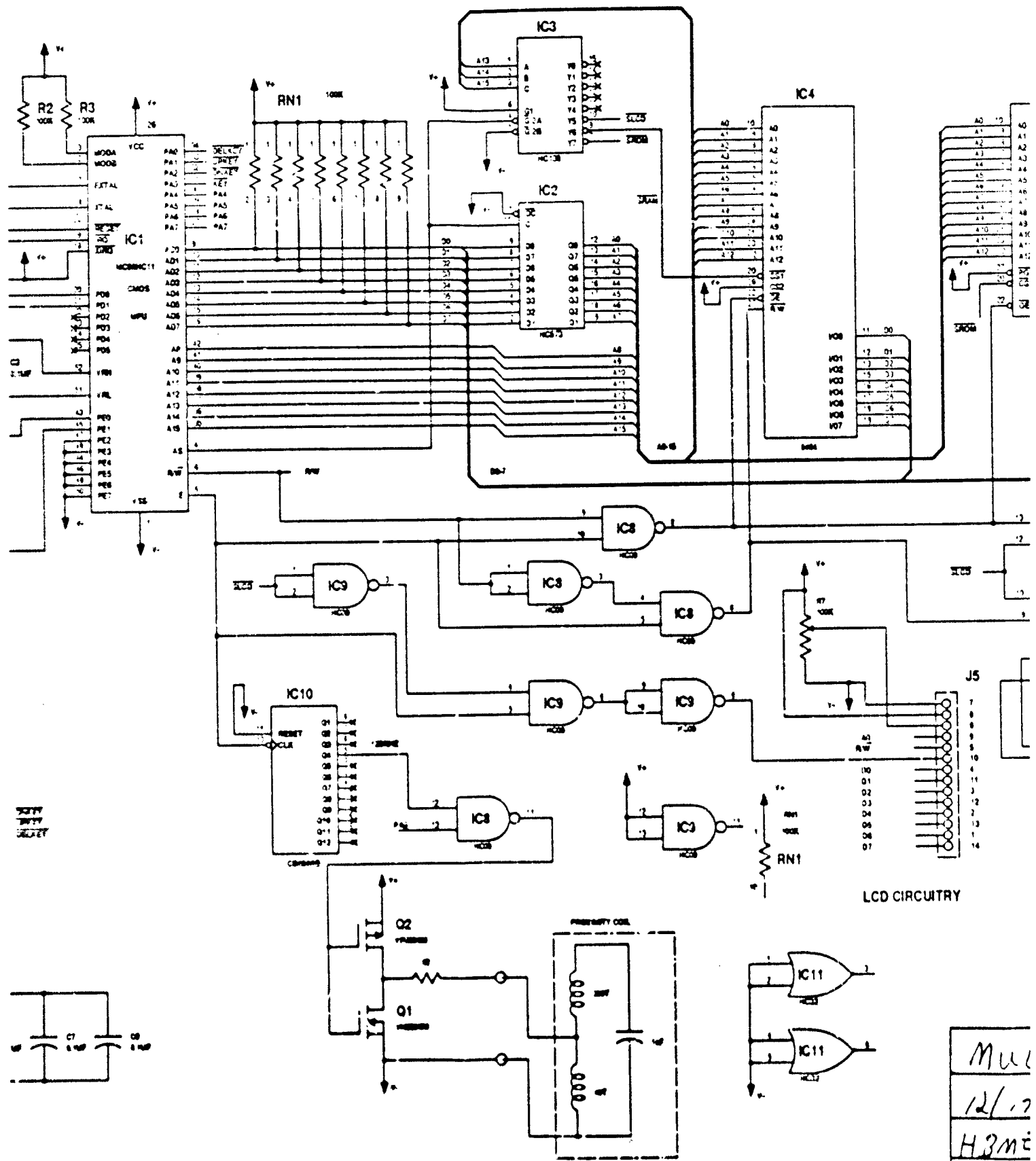
To add a second casualty, the multimonitor is held over the second PMC, and the ADD button pressed. Data from both subjects are then displayed. Likewise, additional subjects can be added, and their status displayed on the screen, up to a maximum of 10.

To remove a subject from the group being monitored, it is necessary only to move the cursor up or down by pressing the up or down arrow buttons. When the cursor is on the subject to be deleted, pressing the DELETE button will remove that casualty from the monitor. To obtain operating instructions for the multimonitor it is only necessary to press the HELP button.

The multimonitor operates on standard "D" cell batteries. It does not have a lighted display for use in low light conditions. It does have the check wrist strap, lost communication, low pulse, motion, and low-battery display categories of the HHM.

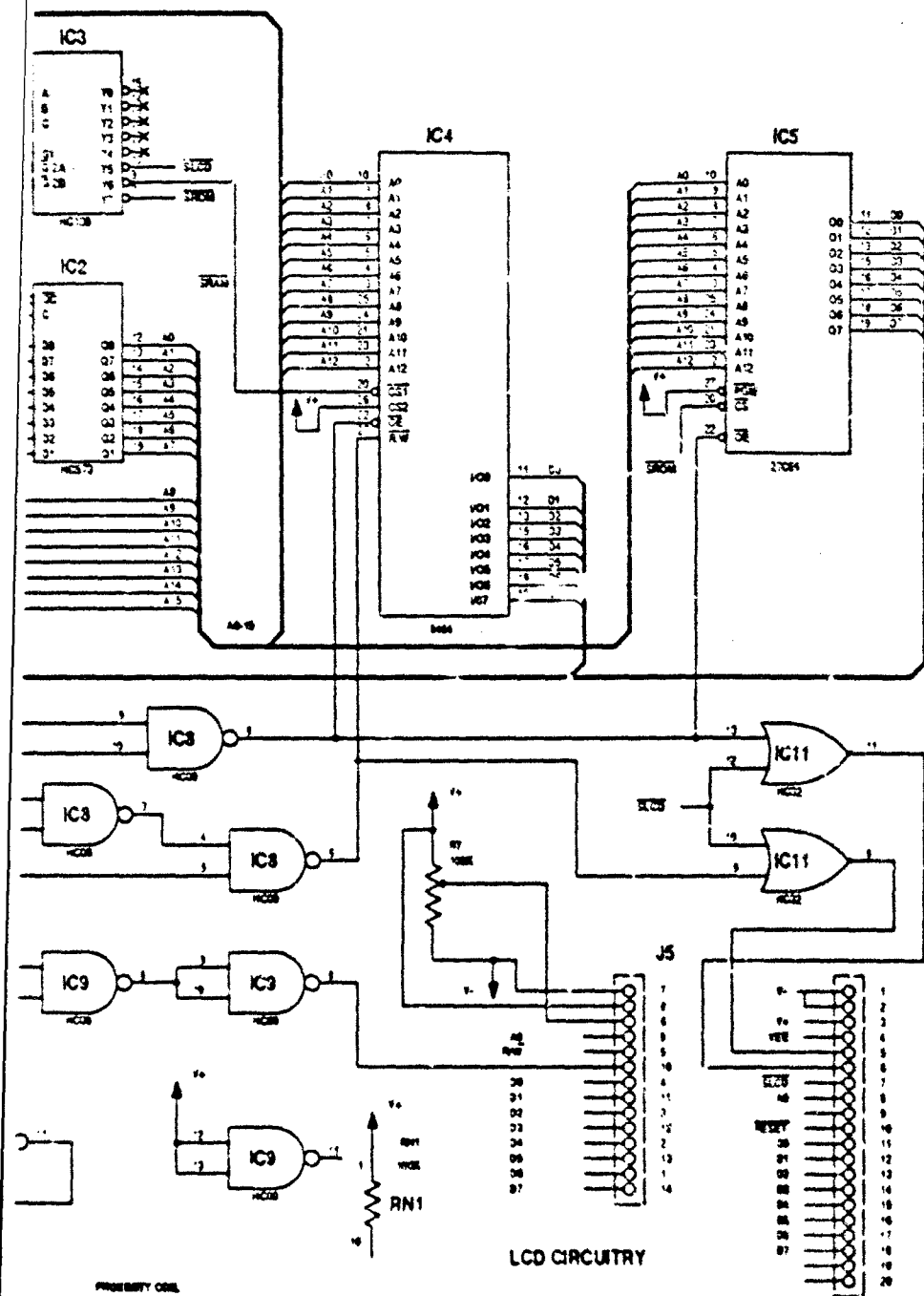
The circuit diagram for the multimonitor is shown in Figure 18. The programming software, which is not included in the Technical Data package, follows the Technical Data Package.





LCD CIRCUITRY

MU
12/17
H3m



MULTIMONITOR SCHEMATIC
 12/17/84 JJ
 HRMEC PURDUE UNIVERSITY

Figure 18

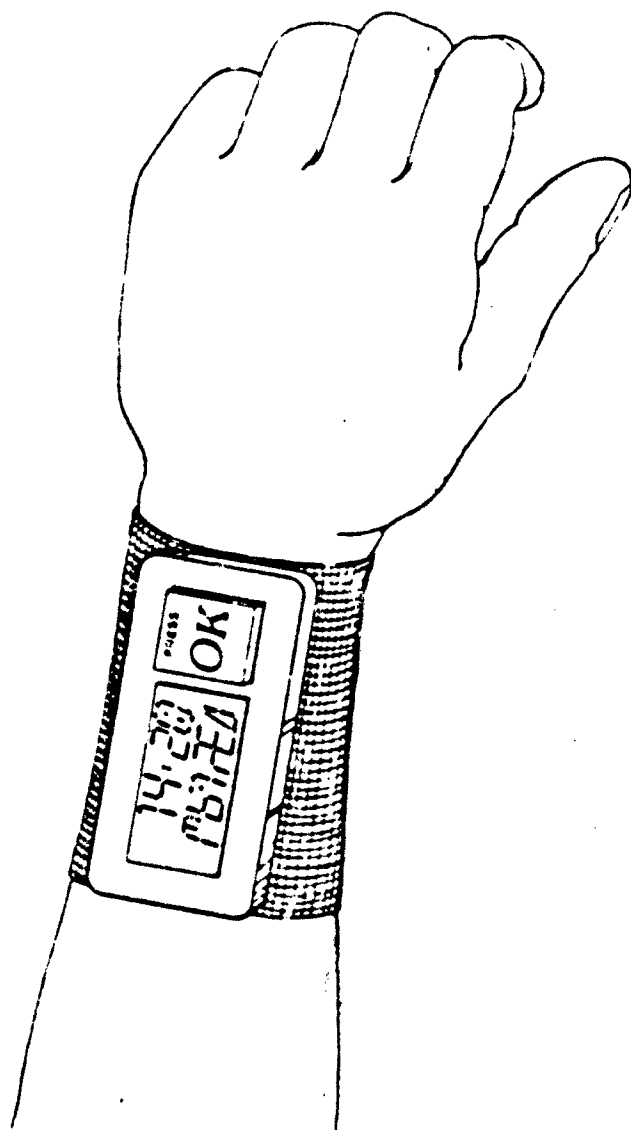
Conclusions: Assessment of Characteristics and Future Enhancements for the Life Detector (Personal Monitor and Communicator)

Some characteristics of the system delivered to Ft. Detrick were different than envisioned at the start of the project, and deserve comment.

We were able to make the hand held monitor smaller than originally expected, and could have made it even smaller, except for the extensive R.F. shielding required by the radio design of the subcontractor (Magnavox) and because of limited visibility of smaller LCD display screens. The wrist-worn unit was lighter in weight than we had anticipated because we were able to use very lightweight materials to make the case. The size of the wrist-worn unit was, however, larger than we had hoped for, due mainly to the size and, again, the shielding requirements of the radios. The radio chosen for the system used a frequency assigned to us by the Army during our previous development effort, and proved to be a substantial problem for both miniaturization and broadcast range of the system. Certainly, the next generation models should use a higher frequency in order to achieve size reduction (and/or a longer range if desired). Selection of a more practical frequency, combined with use of custom chips in the circuitry, should provide a several-fold reduction in size of both the wrist-worn and hand-held units. Size reduction by using custom chips was not possible for the field-test units we delivered due to the limitation of financial resources allocated to the contract by the army. Custom chip design and fabrication, plus other size reduction technology, requires larger volume production to be cost-effective. We are of the opinion that the complete PMC can be incorporated into the wrist-band in the future if adequate resources are deployed toward the effort. The wrist unit would, however, probably be somewhat larger than the artists depiction shown in the Figure 19, which is from our 1986 report.

The multimonitor was substantially ruggedized by us, even through the Army had not requested this ruggedization. We were of the opinion that this improvement was important, since the multimonitor might be used for briefings or laboratory demonstrations at a later date, and therefore it would be useful to have the breadboard look more or less like a fieldable unit.

Once the design was finalized for the field test units, sensing and signal processing showed the sensitivity of the system to detect motion or pulses to be excellent, as we had expected, but we were sometimes unable to differentiate between arterial pulses and some other signals. For example, rapid respiration, which can approach 30 to 60/minute in severely wounded casualties could not be differentiated from bradycardias of 30 to 60 minute by signal processing, including power spectrum analysis. Other periodic mechanical events which are of the same frequency as the heart rates of interest were sometimes troublesome, also. Several man-months of effort were devoted in an attempt to improve the system's rejection of these signals. In our opinion, additional refinement, if possible, would require a major signal-processing effort. In order to remain on schedule we had to proceed with finalizing the design.



Artist's concept of wrist unit with size reductions.

Figure 19.

References

No publications have been submitted or published due to potential patent considerations pursuant to this contract.

TECHNICAL DATA PACKAGE

FOR THE

PERSONAL MONITOR AND COMMUNICATOR

Prepared for:

**Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD**

Prepared by:

**Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47906**

OPERATING AND MAINTENANCE INSTRUCTIONS
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A002

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

OPERATING AND MAINTENANCE INSTRUCTIONS

FOR THE

PERSONAL MONITOR AND COMMUNICATOR

Prepared for:

**Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD**

Prepared by:

**Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47906**

CONTENTS

PREFACE	4
DESCRIPTION.....	5
System.....	5
Personal Monitor Communicator.....	5
Hand Held Monitor.....	5
INSTALLATION.....	9
Personal Monitor Communicator.....	9
Hand Held Monitor.....	9
OPERATION	13
Overview.....	13
Personal Monitor Communicator.....	13
Hand Held Monitor	13
STORAGE	20
Personal Monitor Communicator.....	20
Hand Held Monitor	20
THEORY.....	21
TROUBLESHOOTING GUIDE.....	22
CALIBRATION PROCEDURES.....	22

CONTENTS - continued

FIGURES

1.	Personal monitor communicator	6
2.	Hand held monitor.....	7
3.	System operation	8
4.	Insertion of PMC power supply batteries.....	10
5.	Insertion of HHM power supply batteries.....	11
6.	Insertion of HHM background lighting battery	12
7.	Proper position of PMC electrodes.....	14
8.	System activation	19

PREFACE

This manual contains information needed for the operation and maintenance of a prototype system which provides a means for determining the heart rate of a soldier at rest. This system has the capability of monitoring the status of a soldier at MOPP-4 without touching the soldier or invading the soldier's chemical defense ensemble. The instructions explain the installation, operation, and troubleshooting procedures which can be performed without removing any device covers.

DESCRIPTION

System

The system consists of two components: the Personal Monitor Communicator (PMC) which is worn on the wrist (Figure 1) and the Hand Held Monitor (HHM) which is carried by the medic (Figure 2). The PMCs are units worn by soldiers being monitored. Signals from the stainless steel electrodes in the wristband of each PMC are used to determine the heart rate of the soldier and/or whether the soldier is moving. Radio communication links allow the HHM to display the heart rate to medics who are located up to 5 feet away. The system is operational under conditions of conventional or chemical battlefield warfare at all levels up to and including MOPP-4. See Figure 3.

Personal Monitor Communicator

The PMC is composed of a rectangular box connected by wire leads to a wristband. The wristband contains the tetrapolar dry-electrode impedance measuring plethysmograph necessary for monitoring the soldier. The band wraps around the wrist of the soldier and must be in direct contact with the skin. The box contains a 2-way radio link and a microprocessor. The unit is powered by batteries which fit in a compartment on the side of the box. The box needs to be affixed to the forearm (left or right) of the soldier. There is no need for the box to be in direct contact with the skin. The entire PMC assembly can be worn under a chemical defense jacket or other clothing. See Figure 1.

Hand Held Monitor

The HHM consists of a single rectangular box with a removable antenna. Located on the front face of the HHM are two switches (*ON-OFF* switch and *ACTIVATE* button) and a display panel. The *ON-OFF* switch controls the power to the HHM. The *ACTIVATE* button causes the HHM to gather information from a specified PMC. The rectangular liquid crystal display (LCD) panel displays the status of the soldier being monitored and the status of the PMC. An additional switch is located on the side of the HHM nearest the *ON-OFF* switch. This switch turns on the background light for night time use of the LCD. Two battery compartments are located on the HHM. The battery compartment located on the top of the unit near the antenna provides power for background lighting of the liquid crystal display. The battery compartment on the bottom of the HHM is the main power supply. The bottom battery compartment cover can be seen in Figure 3.

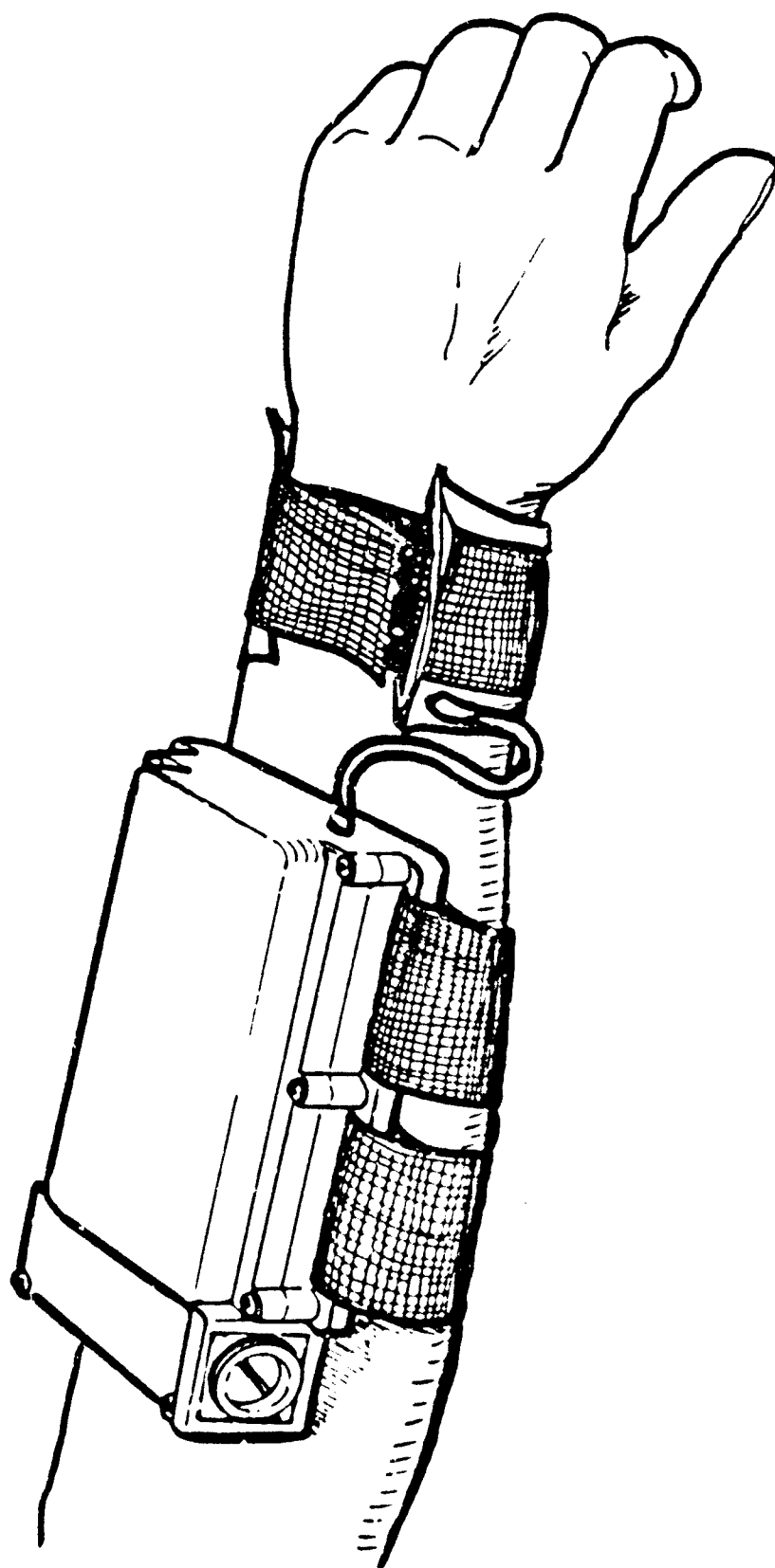


FIGURE 1. *Personal monitor communicator.*

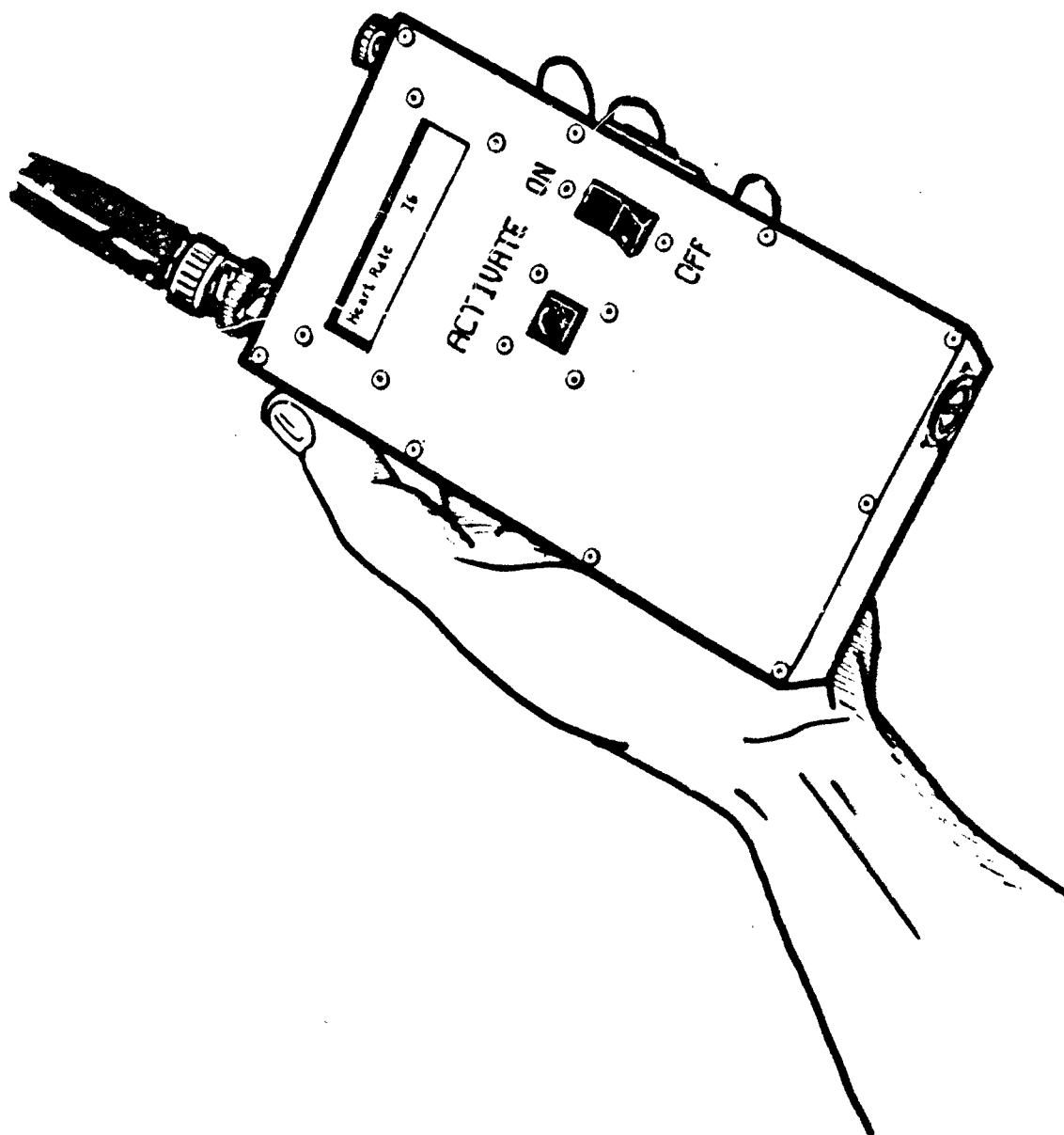


FIGURE 2. *Hand held monitor.*

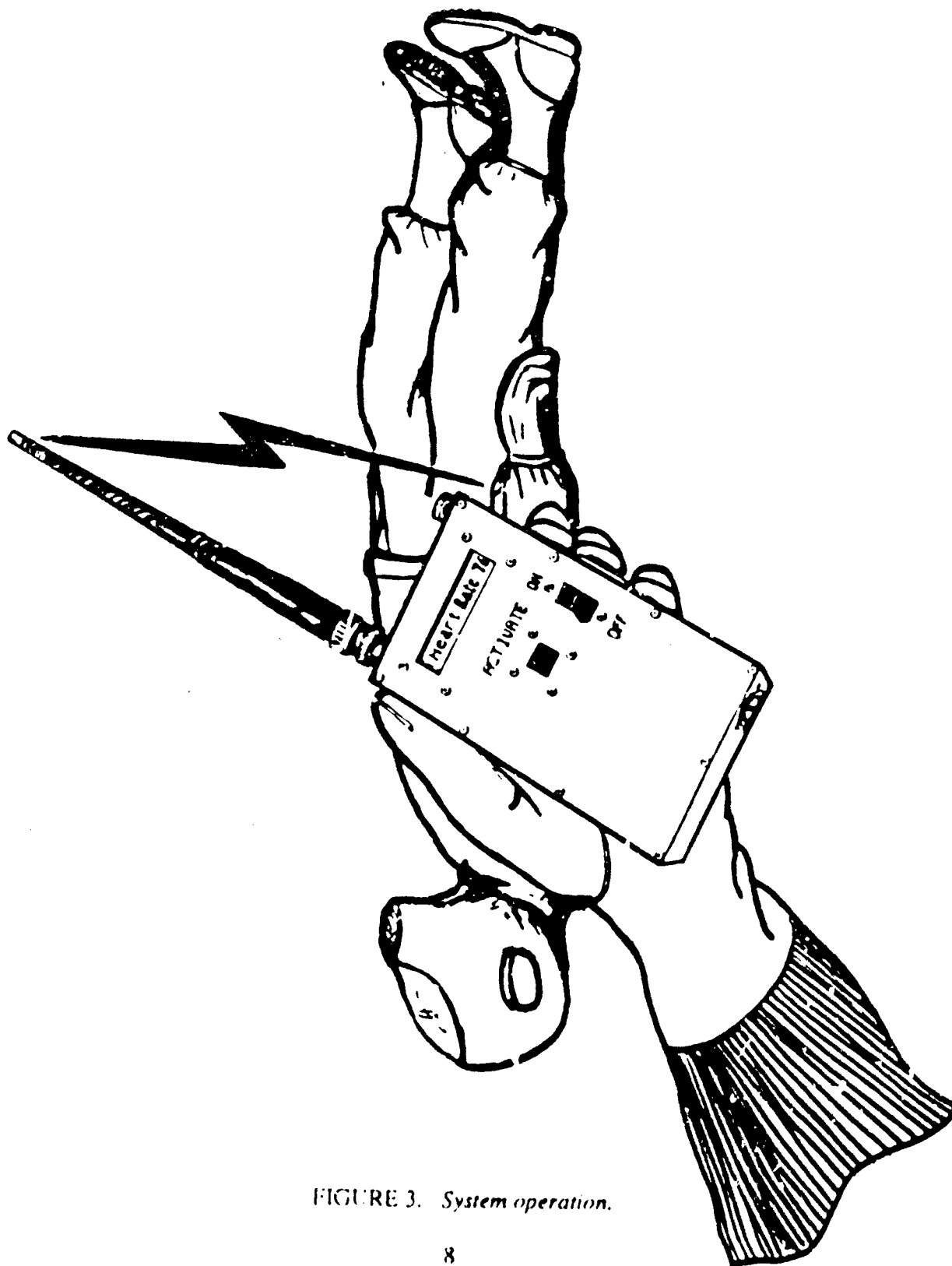


FIGURE 3. *System operation.*

INSTALLATION

Personal Monitor Communicator

The PMC requires *two 1/2 A 3-volt lithium batteries* for the unit to be operable. The lithium batteries will supply approximately 12 hours of power to the PMC when the PMC is in continuous use. When the PMC is in standby mode (not communicating with an HHM), the batteries will last for 4 weeks.

The battery compartment is located on the side of the rectangular box. The compartment cover can be removed by unscrewing the brass cover with any coin or key. Insert one battery at a time. The positive end (bump-end) of each battery *must* always be the first end to enter the compartment. Replace the compartment cover. See Figure 4. *Do not over tighten the brass cover.*

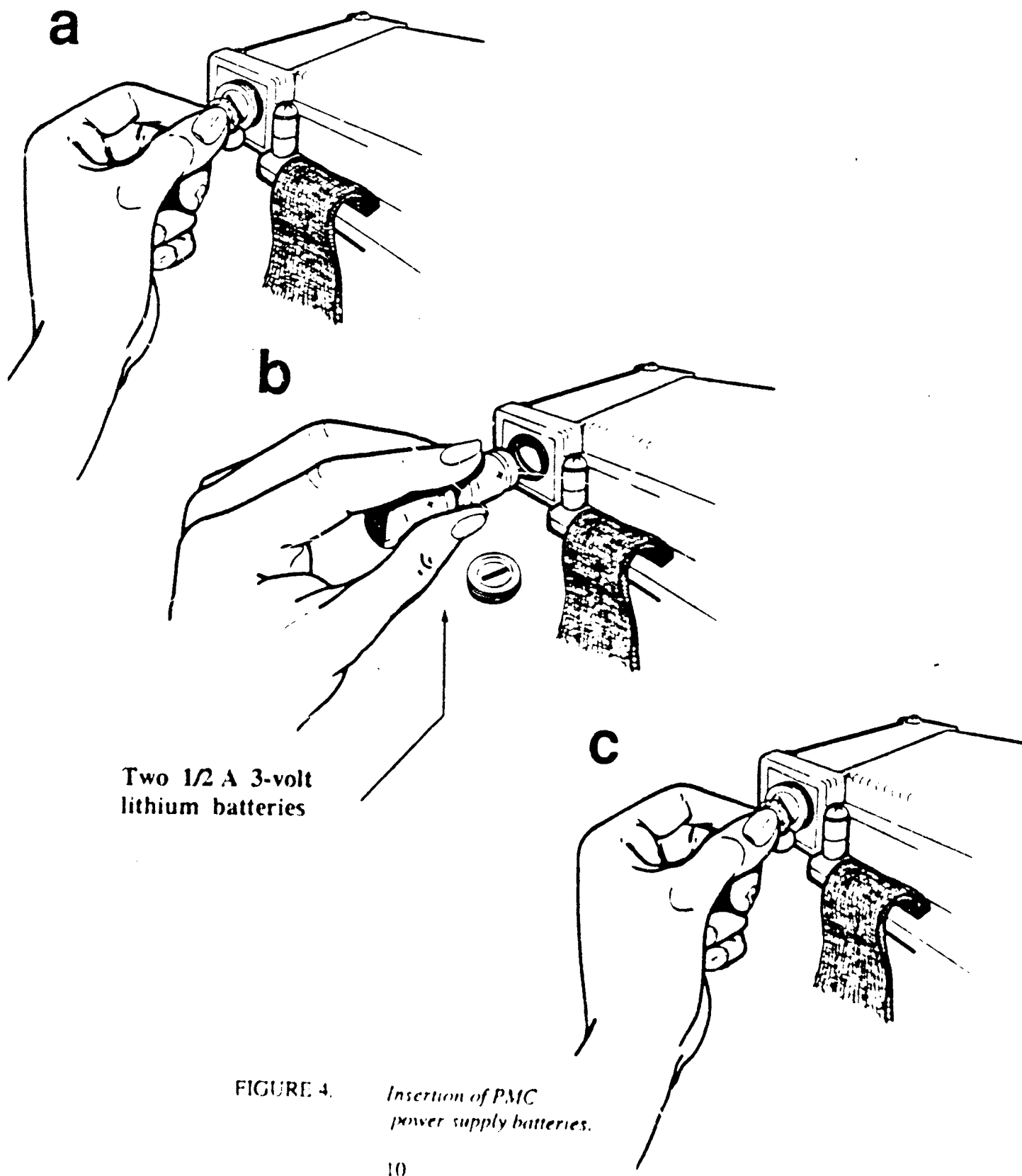
Hand Held Monitor

The HHM has a detachable antenna which connects to the BNC connector on top of the HHM. The antenna must be mounted when the HHM is in use. With the antenna, the HHM can monitor a PMC up to 5 feet away.

The HHM requires *two 2/3 A 3-volt lithium batteries* for the unit to be operable and *one 9-volt alkaline battery* to provide background lighting for the LCD. The lithium batteries will supply approximately 12 hours of power to the HHM when the HHM is in continuous use.

Insert the power supply batteries in the compartment on the bottom of the HHM unit. Remove the compartment cover by unscrewing the brass cover with any coin or key. Insert two 3-volt lithium batteries in this compartment. The positive end (bump-end) of each battery *must* always be the first end to enter the compartment. Replace the compartment cover. See Figure 5. *Do not over tighten the brass cover.*

Background lighting for the LCD is necessary when the hand held monitor is used at night or in low light conditions. The 9-volt alkaline battery provides this lighting. To install the battery, loosen the thumb screws on the compartment cover on the top of the unit near the antenna. Swivel the cover away from the battery compartment. Attach the 9-volt battery to the battery clip. Insert the 9-volt battery into the compartment. Replace the compartment cover and tighten the thumb screws. See Figure 6.



Two 1/2 A 3-volt
lithium batteries

FIGURE 4. *Insertion of PMC
power supply batteries.*

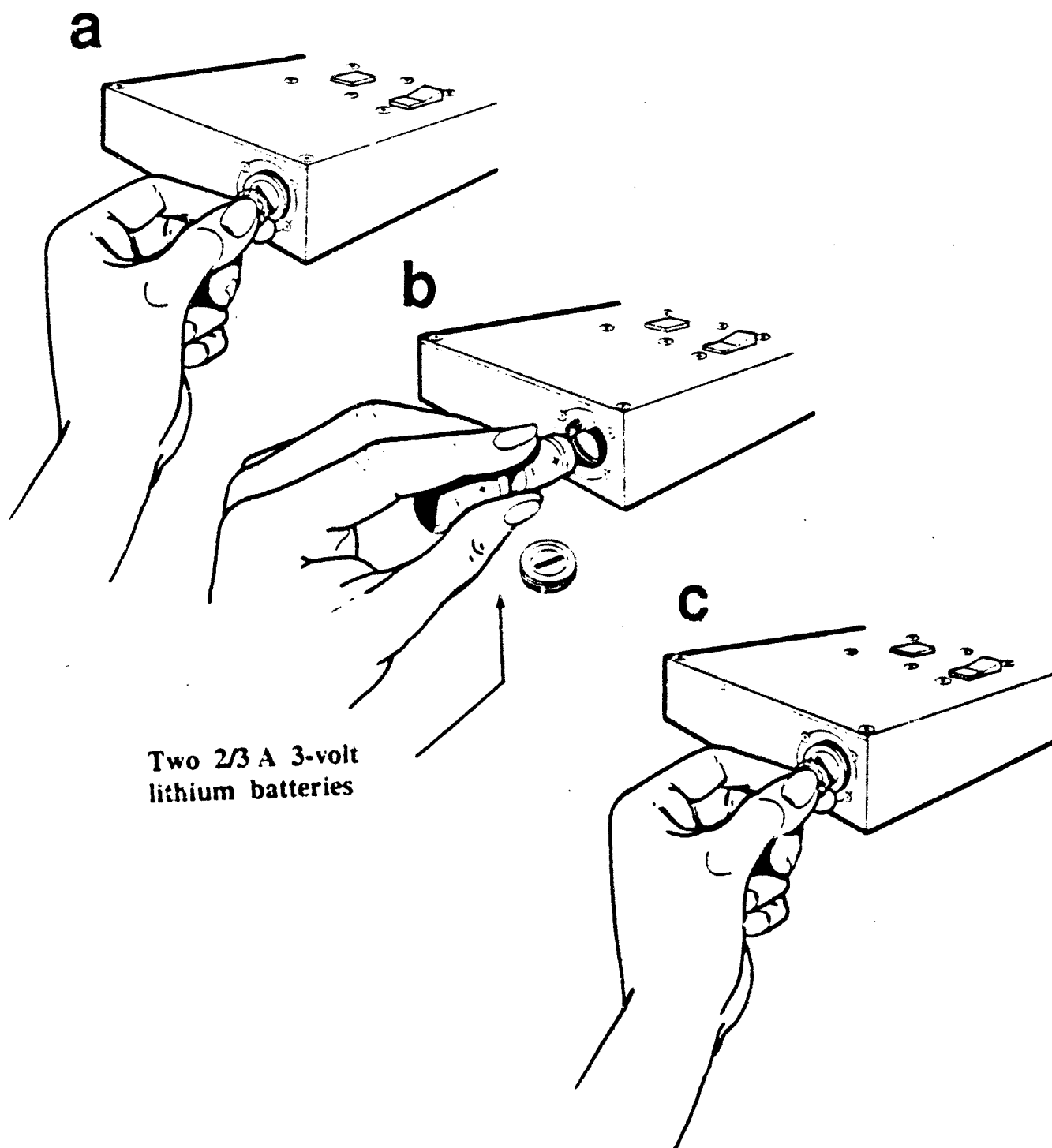


FIGURE 5. *Insertion of HHM power supply batteries.*

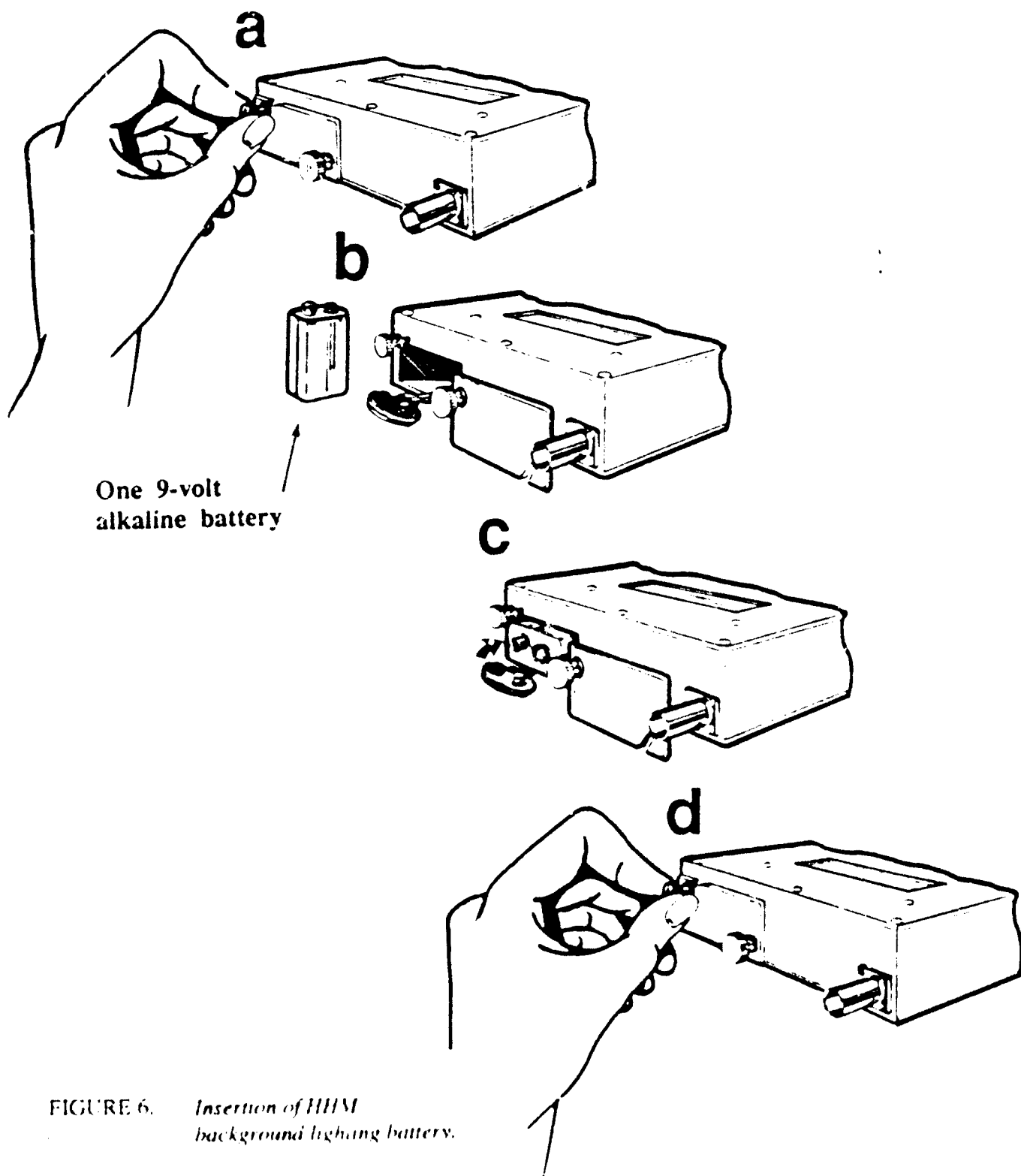


FIGURE 6. *Insertion of HHM background lighting battery.*

OPERATION

Overview

The function of the system is to determine the heart rate of a soldier and/or whether the soldier is moving. The information is gathered from a personal monitor communicator (PMC) worn by the soldier. The PMC must be strapped on correctly and be worn at least 10 to 20 minutes before the soldier can be accurately monitored. The PMC must remain on the soldier as long as the soldier needs to be monitored. The hand held monitor (HHM) provides a display of the heart rate information. The HHM operator is responsible for selecting the PMC to monitor and for understanding the information displayed on the HHM.

Personal Monitor Communicator

Proper monitoring of the soldier depends on the PMC. Carefully follow the instructions given below. Give special attention to assure the wristband is strapped on correctly, as shown in Figure 1.

1. Place the box and wristband on a flat surface with the box closest to you. The box should be placed so both elastic straps are completely visible and lie flat. The wristband should be open with the stainless steel electrodes exposed.
2. Place the palm side of the wrist in the center of the wristband. The wristband should be positioned in the same area as a wristwatch would be worn. Be sure the palm side of the wrist is in contact with the metal electrodes. Secure the wristband to the wrist with the Velcro closures. If fingers start to "tingle," loosen the band slightly. Figure 7 shows the position of the four metal electrodes on the wrist.

Note: If the wristband is applied too tightly, blood flow in the arm will be reduced and the heart rate signal will be lost.

3. Place the rectangular box on the forearm between the wrist and the elbow. Use the two elastic straps and the Velcro closures to secure the box at this location. Make sure the straps are secure yet comfortable.
4. The PMC must remain on the wrist for 10 to 20 minutes before accurate results are possible.

Hand Held Monitor

The HHM displays heart rate information for a soldier when the monitor is correctly used. Follow the instructions given in this section carefully.

Turn the HHM on by pressing the *ON-OFF* switch on the front of the HHM to the *ON* position. If no daylight or artificial light is present, press the button on the side of the HHM to provide background lighting for the LCD.

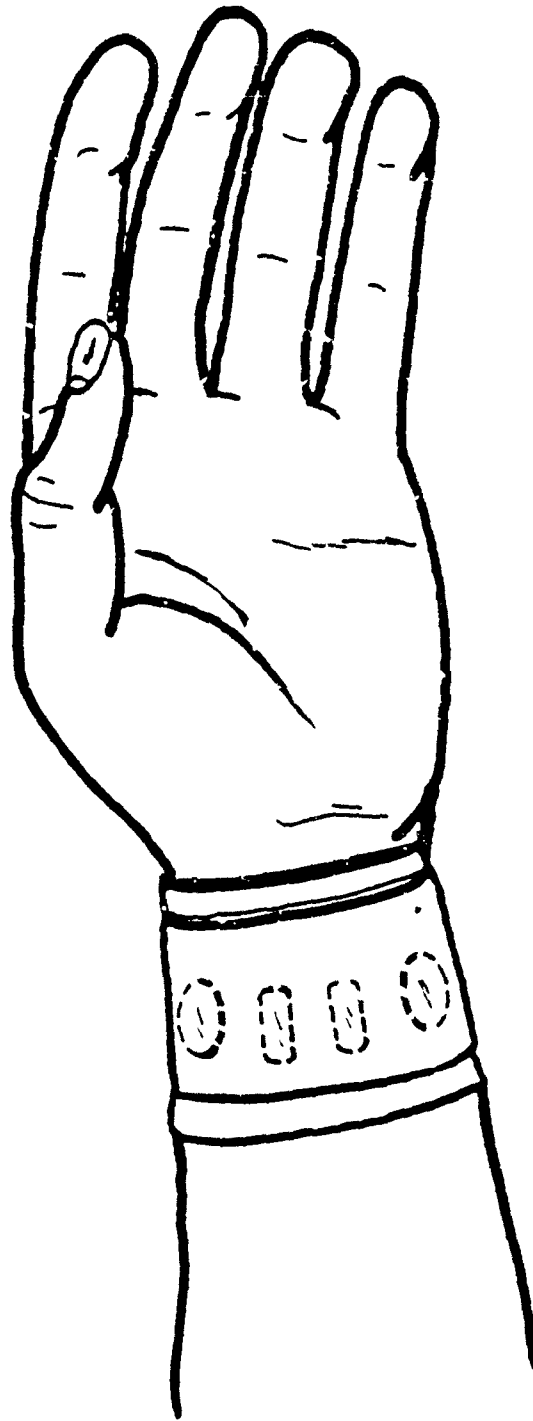


FIGURE 7. *Proper position of PMC electrodes.*

The HHM indicates it is on by displaying the following message:

Hand Held
Monitor (HHM)

This message remains on the screen for only a few seconds.

The HHM then displays the following instructions:

Put HHM Near PMC
Press *ACTIVATE*

Hold the back side of the HHM within 2 inches of the rectangular box of the PMC that is to be monitored. See Figure 8. Leave the HHM in this location and briefly press and release the *ACTIVATE* button on the front of the HHM. The HHM displays the message:

Attempting
PMC Activation

When the activate procedure works, the HHM displays the message:

PMC ACTIVATED

The HHM can now be moved a few feet away from the PMC.

The HHM will attempt to communicate with the designated PMC and will display the message:

Attempting PMC
Communication

When communication is established between the HHM and the PMC, the HHM begins to display messages concerning the soldier's status. If the sensed heart rate is reasonable, then the message:

Heart Rate #

where the character '#' is replaced by the soldier's heart rate, appears.

Other messages that may occur during this phase of monitoring include:

Low Pulse

or

Motion

Messages will be displayed approximately every 15 seconds after the PMC has been activated. If motion is present, the heart rate may not be displayed.

Monitoring a new PMC may begin during any point of the HHM operation. To monitor a new PMC, hold the back side of the HHM approximately 2 inches above the rectangular box of the new PMC and press the *ACTIVATE* button. The HHM now ignores the previously active PMC and monitors the newly selected PMC. More than one HHM may monitor a single PMC.

Conditions may require the HHM software to make several attempts to activate the selected PMC. The retransmission message:

Attempting
Retransmission

will appear on the screen until the procedure fails or the PMC is activated.

If the activate procedure fails, then the following message is displayed:

ERROR--Repeat
Activate Steps

and the activate procedure must be repeated.

After the PMC is activated, several attempts may be made by the HHM to communicate with the PMC. During these attempts, the following message will appear on the LCD screen:

Attempting
Retransmission

This message indicates the HHM has not successfully prompted the activated PMC.

If the HHM is unable to establish communication with the PMC then the HHM displays the message:

Lost
Communication

After this message, the PMC must be re-activated so the HHM displays the message:

Put HHM Near PMC
Press ACTIVATE

If the message:

Check
Wrist Strap

appears, then the PMC is not properly strapped to the soldier. The HHM will not be able to determine the soldier's status when this problem exists.

If the message:

Low PMC Battery

appears, then the batteries in the PMC unit are weak and must be replaced.

If the message:

Low HHM Battery

appears, then the batteries in the HHM unit are weak and must be replaced.

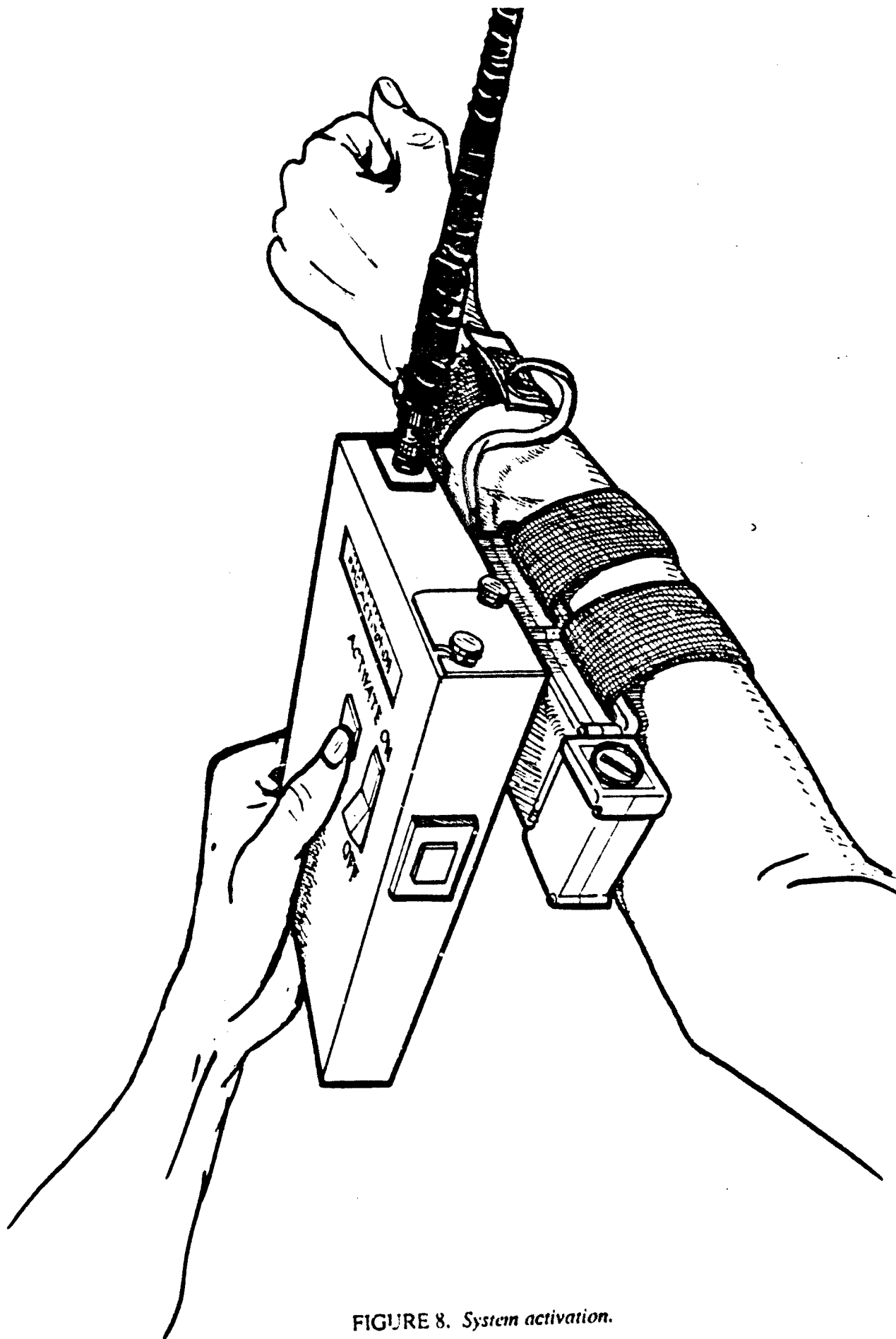


FIGURE 8. System activation.

STORAGE

Personal Monitor Communicator

Wipe the PMC off when it is removed from the soldier's arm. To conserve battery life, remove the batteries from the PMC when it will not be used during the next 24 hour period.

Hand Held Monitor

Turn the HHM off to disengage all radio frequencies. Do this by pressing the *ON-OFF* switch on the face of the HHM to the *OFF* position. The LCD screen is blank when the HHM is turned off. Remove the batteries from the HHM when the unit will not be used during the next 6 month period. Remember to empty the battery compartment located on the top of the unit near the antenna and to empty the battery compartment on the bottom side of the HHM.

THEORY

Heart rate can be measured by monitoring a peripheral pulse. One method of measuring such a pulse is by implementing a plethysmograph. Four electrodes (referred to as a tetrapolar electrode system) are placed around the wrist. As the circulatory system pumps blood through the arteries at the wrist, the impedance changes at the electrodes can be sensed by electronic circuitry. These changes are interpreted as pulses. This method is easy to apply, is non-invasive and is only slightly affected by variations in temperature and barometric pressure.

The wristband of the PMC contains the four electrodes. The device functions properly only when all four electrodes are in full contact with the skin. The wrist impedance signal measured by the plethysmograph is connected to the A/D converter contained in the microprocessor. The processor accesses a software algorithm to analyze the digital signal and compute the heart rate. The resulting value is stored in memory. The real time process continuously updates the heart rate value while the PMC unit is activated.

The heart rate is displayed on the LCD of the HHM when a transmission link exists between the HHM and the PMC. The transmission link is established when the HHM is placed within 2 inches of a selected PMC and the *ACTIVATE* button of the HHM is pressed. This action causes a coil in the HHM to induce a current in a coil in the PMC which in turn activates the PMC. The PMC then begins the data collection and the heart rate calculation. The heart rate is transmitted via a radio frequency transceiver to the HHM and displayed on the LCD. The HHM continues to request information from the selected PMC and the PMC continues to transmit updated heart rate information until the HHM-PMC link is broken. A break occurs when the HHM activates a new PMC or when the HHM is turned off. All data collection and heart rate calculations occur automatically. No outside intervention is necessary.

TROUBLESHOOTING GUIDE

Symptom	Cause
No display on the HHM screen	HHM battery compartment may be empty HHM batteries may be dead HHM unit may not be turned on (check <i>ON-OFF</i> switch)
No communication between HHM and PMC	PMC battery compartment may be empty PMC batteries may be low or dead HHM batteries may be low HHM antenna may not be attached HHM may have been too far from PMC when <i>ACTIVATE</i> button was pressed

CALIBRATION PROCEDURES

This section is not applicable to this manual.

SPECIFICATIONS,
TYPE B
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A003

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

SPECIFICATION NUMBER PMC-1989
CODE IDENT 100-PMC-1989
9 October 1989

NON-COMPLEX ITEM DEVELOPMENT SPECIFICATION

FOR

PERSONAL MONITOR AND COMMUNICATOR (PMC)

AUTHENTICATED BY _____ DATE _____

APPROVED BY _____ DATE _____

9/10/89

TABLE OF CONTENTS

Paragraph	Page
1. SCOPE.....	4
1.1 Scope.....	4
2. APPLICABLE DOCUMENTS.....	4
2.1 Government Documents.....	4
2.2 Non-Government Documents.....	4
3. REQUIREMENTS.....	4
3.1 Item Definition.....	4
3.2 PMC Characteristics.....	5
3.2.1 PMC Performance.....	5
3.2.2 PMC Physical Characteristics.....	5
3.2.2.1 PMC Measurements.....	5
3.2.2.2 PMC Protective Coating.....	5
3.2.2.3 PMC Storage Requirements.....	5
3.2.2.4 PMC Batteries.....	5
3.3 HHM Characteristics.....	5
3.3.1 HHM Performance.....	5
3.3.2 HHM Physical Characteristics.....	6
3.3.2.1 HHM Measurements.....	6
3.3.2.2 HHM Protective Coating.....	6
3.3.2.3 HHM Storage Requirements.....	6
3.3.2.4 HHM Batteries.....	6
4. QUALITY ASSURANCE PROVISIONS.....	6
4.1 General.....	6
4.1.1 Responsibility For Inspection.....	6
4.1.2 Special Tests And Examinations.....	6
4.2 Quality Conformance Inspections.....	7

CONTENTS - continued

Paragraph	Page
5. PREPARATION FOR DELIVERY.....	7
5.1 General.....	7
5.2 Specific Requirements.....	7
5.3 Detailed Preparation	7
5.3.1 Preservation And Packaging.....	7
5.3.2 Packaging	8
5.3.3 Marking For Shipment.....	8
6. NOTES.....	8
6.1 Inteded Use	8

FIGURES

Figure	Page
1. PMC dimensional measurements.....	9
2. PMC wriststrap dimensional measurements.....	10
3. HHM dimensional measurements	11

9/10/89

1. SCOPE

1.1 *Scope.* This specification establishes the performance, design, development, and test requirements for the Personal Monitor and Communicator non-complex item.

2. APPLICABLE DOCUMENTS

2.1 *Government Documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

SPECIFICATIONS

Military

MIL-S-83490 Specifications, Types and Forms

STANDARDS

Military

MIL-STD-129F Marking For Shipment And Storage

MIL-STD-433A Configuration Management Practices For
Systems, Equipment, Munitions, and Computer
Programs

MIL-STD-490A Specifications Practices

2.2 *Non-Government Documents.* This section is not applicable to this specification.

3. REQUIREMENTS

3.1 *Item Definition.* The Personal Monitor and Communicator provides a means of remotely monitoring a soldier's medical status. The non-complex item is comprised of the Personal Monitor Communicator (PMC) and the Hand Held Monitor (HHM). The PMC is an electronic device worn around the wrist of a soldier. The HHM activates PMCs and displays a soldier's heart rate on an LCD display.

3.2 *PMC Characteristics.*

3.2.1 *PMC Performance.* The PMC uses tetrapolar electrodes, which are mounted on a wrist strap, to measure the AC impedance present at the wrist. The heart rate and body motion of the soldier are extracted from the impedance signal through signal processing techniques. This information is transmitted via an RF digital transceiver upon request. The PMC requires two 1/2 A 3-volt lithium batteries for the unit to be operable. The lithium batteries will supply approximately 12 hours of power to the PMC when the PMC is in continuous use. When the PMC is in standby mode (not communicating with an HHM), the batteries will last for 4 weeks.

3.2.2 *PMC Physical Characteristics.*

3.2.2.1 *PMC Measurements.* Each PMC unit, complete with batteries, weighs approximately 248 grams. See Figure 1 and Figure 2 for the dimensional measurements of the unit.

3.2.2.2 *PMC Protective Coating.* The flat olive-drab polyurethane paint (by Pactra, number 20123) serves as a protective coating. The unit has not been waterproofed.

3.2.2.3 *PMC Storage Requirements.* The two lithium batteries should be removed when the PMC is stored.

3.2.2.4 *PMC Batteries.* Recommended batteries for the PMC are two Panasonic BR 1/3 A 3-volt lithium cells.

3.3 *HHM Characteristics.*

3.3.1 *HHM Performance.* The HHM activates a PMC worn by a soldier. The monitor then repeatedly requests and displays the soldier's heart rate upon an LCD display. The communication process continues until the HHM activates a new PMC or the HHM is turned off. The HHM requires two 2/3 A 3-volt lithium batteries for the unit to be operable and one 9-volt alkaline battery to provide background lighting for the LCD. The lithium batteries will supply approximately 12 hours of power to the HHM when the HHM is in continuous use. The alkaline battery will supply approximately 4 hours of backlighting.

9/10/89

3.3.2 *HHM Physical Characteristics.*

3.3.2.1 *HHM Measurements.* Each HHM unit, complete with batteries and antenna, weighs approximately 921 grams. See Figure 3 for the dimensional measurements of the unit.

3.3.2.2 *HHM Protective Coating.* The flat olive-drab polyurethane paint (by Pactra, number 20123) serves as a protective coating. The unit has not been waterproofed.

3.3.2.3 *HHM Storage Requirements.* The three batteries (two lithium and one alkaline) should be removed when the HHM is stored.

3.3.2.4 *HHM Batteries.* Recommended batteries for the HHM are two Panasonic BR 2/3 A 3-volt lithium cells and one 9-volt alkaline cell.

4. QUALITY ASSURANCE PROVISIONS

4.1 *General.*

4.1.1 *Responsibility For Inspection.* Unless otherwise specified in the contract or order, the supplier is responsible for the performance of all inspection requirements as specified herein. Except as otherwise specified, the supplier may utilize his own facilities or any commercial laboratory acceptable to the contracting agency. The contracting agency reserves the right to perform any of the inspections set forth in the specification where such inspections are deemed necessary to assure supplies and services conform to prescribed requirements.

4.1.2 *Special Tests And Examinations.* This section is not applicable to this specification.

9/10/89

4.2 *Quality Conformance Inspection.* Before testing the Personal Monitor and Communicator, install two lithium batteries (1/2 A 3 Volt) in a PMC unit, install two lithium batteries (2/3 A 3 Volt) in the bottom compartment of a HHM unit and install a 9 volt alkaline battery in the top compartment of the HHM unit. Then attach an antenna to the HHM. Begin testing the operation of the devices by switching the HHM power to the *ON* position. Press the switch on the side of the HHM to verify backlighting is being provided to the LCD display of the HHM. Activate the PMC with the HHM to confirm communication exists between the HHM and the PMC. At the end of the test session turn the HHM off. Remove the two batteries from the PMC and the three batteries from the HHM before storing the units.

5. PREPARATION FOR DELIVERY

5.1 *General.* The batteries must be removed from all PMC and HHM units before the units are packed for shipping. In addition, the antennas must be removed from all the HHM units. The PMC and HHM units shall then be securely packed in corrugated cardboard boxes. Avoid exposing the box to extreme heat, extreme cold, or submersion.

5.2 *Specific Requirements.* Each PMC and HHM unit shall be placed in an individual anti-static foil wrap. Then the units and the HHM antennas shall be placed in a sturdy corrugated cardboard box. Plastic bubble wrap will protect the units and antennas. The batteries shall be placed in boxes provided by the battery suppliers. These boxes shall then be packed in the cardboard box.

5.3 *Detailed Preparation.*

5.3.1 *Preservation And Packaging.* All batteries must be removed from the PMC and HHM units before the units can be packed for shipping. Remove the two lithium batteries from the PMC, and remove the two lithium batteries and the alkaline battery from the HHM. Place all lithium batteries in the lithium box provided by the manufacturer. Place the alkaline batteries in the box provided by the manufacturer. All antennas must be removed from the HHM units before the HHMs can be packed. Inspect the units to assure they are clean and dry. The units shall then be placed in anti-static foil wrap. The batteries shall be stored in their own boxes. Plastic bubble wrap shall be placed around all PMC and HHM units and all HHM antennas to prevent any shifting during shipping.

100-PMC-1989

9/10/89

5.3.2 *Packing.* Sturdy corrugated cardboard boxes shall be used and no additional bracing will be necessary. If there is any possibility the box will become submerged in water, then waterproofing of the box shall become necessary.

5.3.3 *Marking For Shipment.* The boxes containing the PMC and HHM units shall have the markings "FRAGILE" and "ELECTRONIC EQUIPMENT."

6. NOTES

6.1 *Inteded Use.* This specification is to be used to establish the performance, design, development, and test requirements for the Personal Monitor and Communicator non-complex item.

100-PMC-1989

9/10/89

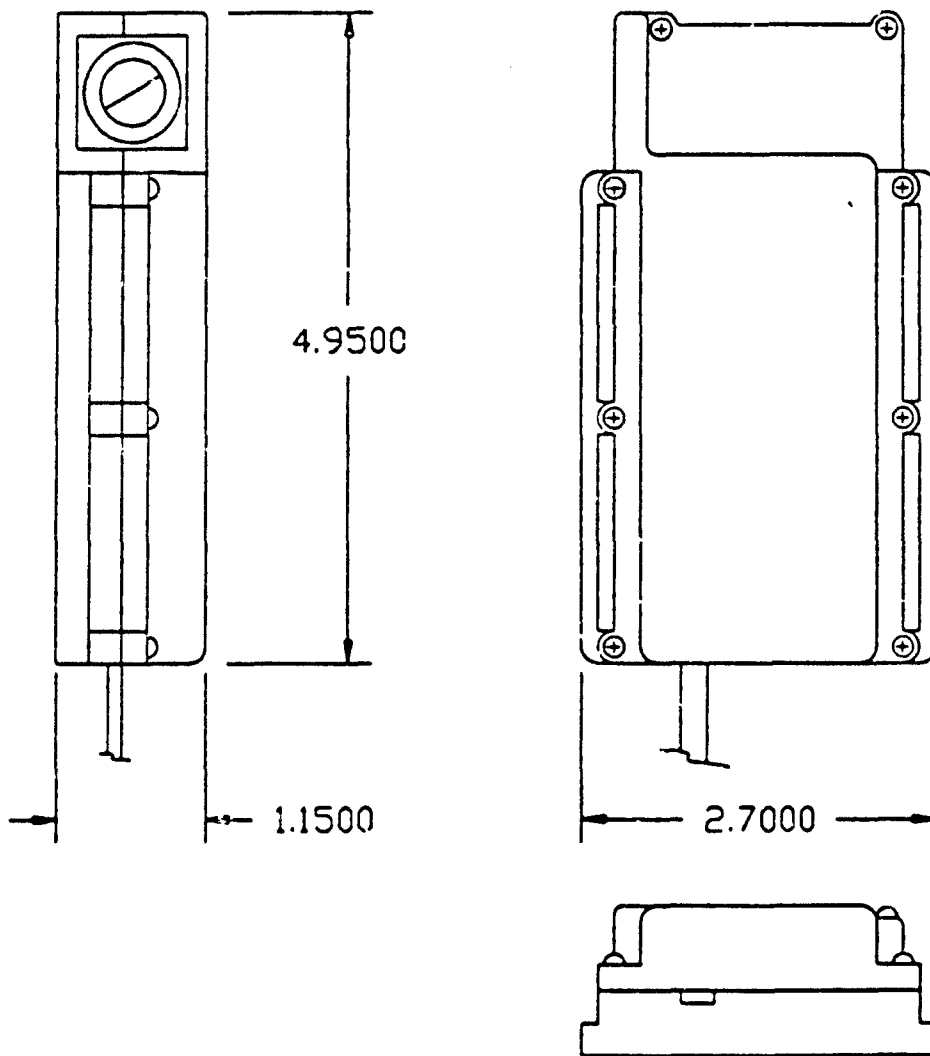


FIGURE 1. *PMC dimensional measurements.*

100-PMC-1989

9/10/89

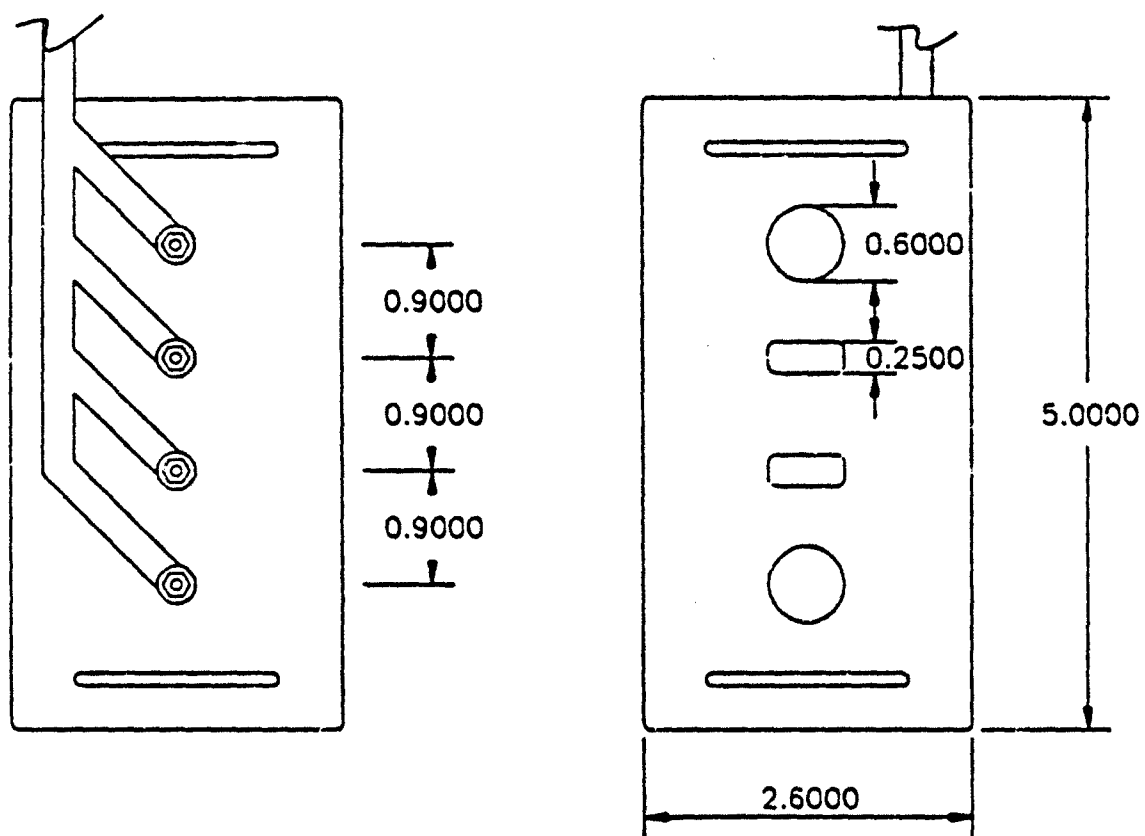


FIGURE 2. *PMC wriststrap dimensional measurements.*

9/10/89

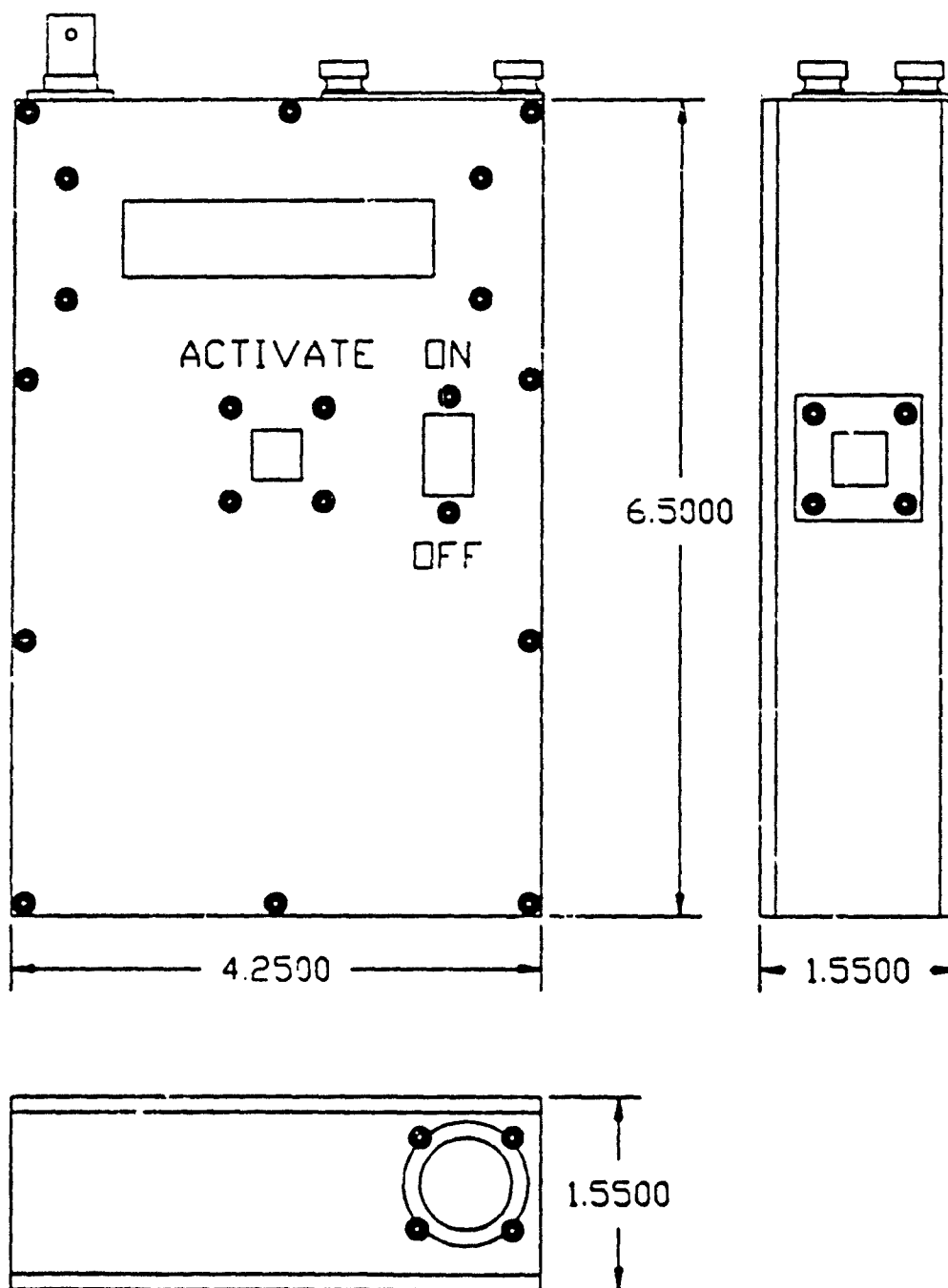


FIGURE 3. *HHM* dimensional measurements.

SPECIFICATIONS,
TYPE B

FOR THE

PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A004

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

A004

See sections

A005 SOFTWARE REQUIREMENTS SPECIFICATIONS

and

A006 INTERFACE REQUIREMENTS SPECIFICATIONS

PART I OF TWO PARTS

SOFTWARE REQUIREMENTS SPECIFICATION
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A005

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

CONTENTS

Paragraph	Page
1. SCOPE.....	7
1.1 Identification	7
1.2 Purpose	7
1.3 Introduction	7
2. APPLICABLE DOCUMENTS.....	7
2.1 Government Documents	7
2.2 Non-Government Documents	7
3. REQUIREMENTS.....	8
3.1 Programming Requirements.....	8
3.1.1 Programming Language(s)	8
3.1.2 Compiler/Assembler	8
3.1.3 Programming Standards	8
3.2 Design Requirements.....	8
3.2.1 Sizing and Timing Requirements	8
3.2.2 Design Standards	9
3.2.3 Design Constraints	9
3.3 Interface Requirements.....	9
3.3.1 Interface Relationships.....	9
3.3.2 Interface Identification and Documentation	9
3.3.3 Detailed Interface Requirements	9
3.3.3.1 CSCI-to-CSCI Interface Requirements.....	9
3.3.3.2 CSCI-to-HWCI or Critical Item Requirements.....	9
3.3.3.2.1 Wrist Impedance Circuit Interfaces	9
3.3.3.2.2 Proximity Receiver Interface.....	10
3.3.3.2.3 Voltage Reference Interface	10
3.3.3.2.4 Digital RF Transceiver Interfaces	10
3.3.3.2.5 Real Time Clock Interface	11
3.4 Detailed Functional and Performance Requirements.....	12
3.4.1 Power Up Initialization Function	12
3.4.1.1 Inputs.....	12
3.4.1.2 Processing	12
3.4.1.3 Outputs.....	12

CONTENTS - continued

Paragraph		Page
3.4.2	Main Monitor Function	13
3.4.2.1	Inputs.....	13
3.4.2.2	Processing	13
3.4.2.3	Outputs.....	14
3.4.3	Proximity Interrupt Function.....	14
3.4.3.1	Inputs.....	14
3.4.3.2	Processing	14
3.4.3.3	Outputs.....	15
3.4.4	Clock Interrupt Function	15
3.4.4.1	Time Keeping Function.....	15
3.4.4.1.1	Inputs	15
3.4.4.1.2	Processing	15
3.4.4.1.3	Outputs.....	15
3.4.4.2	A/D Conversion Function	16
3.4.4.2.1	Inputs	16
3.4.4.2.2	Processing	16
3.4.4.2.3	Outputs.....	16
3.4.5	Serial Receive Interrupt Function.....	17
3.4.5.1	Packet Construction Function	17
3.4.5.1.1	Inputs	17
3.4.5.1.2	Processing	17
3.4.5.1.3	Outputs.....	17
3.4.5.2	Input Packet Processing Function.....	17
3.4.5.2.1	Inputs	18
3.4.5.2.2	Processing	18
3.4.5.2.3	Outputs.....	18
3.4.6	Get Data Function	18
3.4.6.1	Inputs.....	18
3.4.6.2	Processing	19
3.4.6.3	Outputs.....	19
3.4.7	Get Rates Function	19
3.4.7.1	Preprocessor Function.....	19
3.4.7.1.1	Inputs	19
3.4.7.1.2	Processing	19
3.4.7.1.3	Outputs.....	20

CONTENTS - continued

Paragraph		Page
3.4.7.2	Heart Rate Calculations Function	20
3.4.7.2.1	Inputs	20
3.4.7.2.2	Processing	20
3.4.7.2.3	Outputs.....	21
3.4.8	Transmit Function	21
3.4.8.1	Inputs.....	21
3.4.8.2	Processing	21
3.4.8.3	Outputs.....	22
3.5	Adaptation Requirements.....	22
3.5.1	System Environment	22
3.5.2	System Parameters.....	22
3.5.3	System Capacities.....	22
3.6	Quality Factors.....	22
3.6.1	Correctness Requirements	22
3.6.2	Reliability Requirements	22
3.6.3	Efficiency Requirements	22
3.6.4	Integrity Requirements.....	22
3.6.5	Usability Requirements.....	22
3.6.6	Maintainability Requirements	23
3.6.7	Testability Requirements.....	23
3.6.8	Flexibility Requirements.....	23
3.6.9	Portability Requirements	23
3.6.10	Reusability Requirements	23
3.6.11	Interoperability Requirements	23
3.6.12	Additional Quality Factor Requirements	23
3.7	CSCI Support.....	23
3.7.1	Facility Requirements.....	23
3.7.2	Equipment Requirements	23
3.7.3	Software Requirements.....	24
3.7.4	Personnel Requirements	24
3.8	Traceability	24
4.	QUALIFICATION REQUIREMENTS	24
4.1	General Qualification Requirements	24
4.2	Special Qualification Requirements.....	24

CONTENTS - continued

Paragraph	Page
5. PREPARATION FOR DELIVERY	25
5.1 Preparation For Delivery.....	25
6. NOTES.....	25
6.1 Acronyms	25

FIGURES

Figure	Page
1. Interface block diagram.....	26
2. Function block diagram	27
3. Function main monitor flowchart.....	28
4. Function packet construction finite state machine	29
5. Function input packet processing flowchart	30
6. Function preprocessor flowchart.....	31
7. Function heart rate flowchart	32
8. PMC heart rate versus hand calculated heart rate, first database	66
9. Error histogram, first database	67
10. PMC heart rate versus hand calculated heart rate, second database	68
11. Error histogram, second database.....	69

TABLES

Table	Page
1. Interface identification/documentation	34
2. Interface summary	35
3. Request for soldier status (CSS) format	36
4. PMC positive acknowledge (PPA) format	37
5. PMC soldier status (PSS) format	38
6. Function power up initialization input.....	39

CONTENTS - continued

TABLES

Table	Page
7. Function power up initialization output.....	40
8. Function main monitor input.....	43
9. Function main monitor output.....	44
10. Function proximity interrupt input.....	45
11. Function proximity interrupt output.....	46
12. Function time keeping input	47
13. Function time keeping output	48
14. Function A/D conversion input.....	49
15. Function A/D conversion output.....	50
16. Function packet construction input.....	51
17. Function packet construction output.....	52
18. Function input packet processing input.....	53
19. Function input packet processing output	54
20. Function get data input.....	55
21. Function get data output	56
22. Function preprocessor input	57
23. Function preprocessor output.....	58
24. Function heart rate calculation input.....	59
25. Function heart rate calculation output.....	60
26. Function transmit input.....	61
27. Function transmit output.....	62
28. Heart rate codes.....	63
29. Acronym summary	64

APPENDIXES

Appendix	Page
Appendix I	
10. PMC Heart Rate Processing Test	65
Appendix II	
20. PMC Source Code	70

1. SCOPE

1.1 *Identification.* This Software Requirements Specification establishes the requirements for the CSCI identified as Personal Monitor Communicator Software, PMC-CSCI.1, of the Personal Monitor and Communicator (PMC), PMC-SYS.1, System.

1.2 *Purpose.* The Personal Monitor and Communicator System provides a means of remotely monitoring the heart rate and body motion of a soldier. The PMC-CSCI.1 drives the personal monitor communicator which consists of an electronic device worn around the wrist of the soldier. The unit consists of tetrapolar electrodes, a 2-way radio link and a microprocessor-based signal processor. The software collects and processes analog signals received from the electrodes and transmits the processed data when prompted.

1.3 *Introduction.* This document provides detailed and complete specifications of the software developed to drive the personal monitor communicator unit of the Personal Monitor and Communicator System.

2. APPLICABLE DOCUMENTS

2.1 *Government Documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

SPECIFICATIONS

Military

MIL-S-83490 Specifications, Types and Forms

STANDARDS

Military

DOD-STD-2167 Defense System Software Development

MIL-STD-483A Configuration Management Practices For
Systems, Equipment, Munitions, and Computer
Programs

MIL-STD-490A Specifications Practices

2.2 *Non-Government Documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the

contents of this specification shall be considered a superseding requirement.

OTHER PUBLICATIONS

Oppenheim, A.V., Schafer, R.W.
Digital Signal Processing
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975

Kernighan, B.W., Ritchie, D.M.
The C Programming Language
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1978

Introl C MS-DOS Host Guide Version 1.0
Introl Corporation, Milwaukee, WI, 1987

Motorola Semiconductor Technical Data
Advance Information for the MC68HC11A8
HCMOS Single-Chip Microcomputer
Order number MC68HC11A8/D

Programmer's Reference Manual
M68HC11 HCMOS Single-Chip Microcomputer
Order number M68HC11PM/AD

3. REQUIREMENTS

3.1 *Programming Requirements.*

3.1.1 *Programming Languages.* Two programming languages are implemented for this CSCI. The languages are C and Motorola 68HC11 assembler.

3.1.2 *Compiler/Assembler.* The C compiler is written by Introl Corporation and is version C-11 of the Introl-C Cross-Compiler Systems series.

3.1.3 *Programming Standards.* Structured programming techniques are followed.

3.2 *Design Requirements.*

3.2.1 *Sizing and Timing Requirements.* The CSCI uses all of the available 8K bytes of memory provided by the RAM. In addition, the CSCI currently uses 8585 bytes of the 16K bytes supplied by the PROM. Approximately four kilobytes of memory remain in

the PROM for expansions and improvements. The CSCI processes incoming A/D data at a rate of 25 samples/second. Ten seconds are required for data acquisition and 5 seconds are necessary for soldier evaluation. The CSCI also processes incoming serial communication data at a data rate of 300 baud.

3.2.2 Design Standards. Development of this CSCI follows the design standards of the Hillenbrand Biomedical Engineering Center.

3.2.3 Design Constraints. Several constraints are encountered by using the 68HC11 microcomputer. The FFT required for signal processing must implement integer math and the memory is limited.

3.3 Interface Requirements.

3.3.1 Interface Relationships. See interface block diagram (Figure 1).

3.3.2 Interface Identification and Documentation. See interface identification table (Table 1).

3.3.3 Detailed Interface Requirements.

3.3.3.1 CSCI-to-CSCI Interface Requirements. This section is not applicable to this specification.

3.3.3.2 CSCI-to-HWCI or Critical Item Requirements.

3.3.3.2.1 Wrist Impedance Circuit Interfaces.

- a. **Signal Direction.** See interface table (Table 2).
- b. **Signal Format.** The wrist impedance signal interface (WZCA IF) transmits an A/D signal. The off-wrist signal interface (WZCB IF) transmits an 8 bit number ranging from 0 to 255. A signal with a value of 0 indicates 0 volts dc and a signal with a value of 255 indicates 4.5 to 6 volts dc. The power signal interface (WZCC IF) sends a digital output bit (1 = on) to power the wrist impedance circuit. This output bit is bit 4 of Port A of the 68HC11.
- c. **Memory Buffer And Location.** The WZCA interface requires two memory buffers. The first buffer consists of 500 bytes located at variable name "pbuf." The second buffer has 2 bytes located at variable name "npbuf."
- d. **Transfer Protocol.** A value of 0x10 (0x indicates the number to follow is hexadecimal) is sent to the A/D control register, H11ADCTL. When bit 7 of the register goes high (1) register H11ADR1 will contain wrist impedance

information and register H11ADR2 will contain off-wrist information.

- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.2 PRC Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* When the proximity receiver is activated by the HHM, an active low digital signal is transmitted to the interrupt line (IRQ bar) of the 68HC11.
- c. *Memory Buffer And Location.* The interface requires enough memory to accommodate the stack space necessary to handle the interrupt signal sent from the proximity receiver to the CSCI.1.
- d. *Transfer Protocol.* An asynchronous interrupt is used by the proximity receiver to communicate to the CSCI.1.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.3 Voltage Reference Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The signal will have a numerical value in the range of 0 to 255. Signals with values greater than or equal to 68 indicate the HHM batteries are low (< 4.5 volts).
- c. *Memory Buffer And Location.* This section is not applicable to this interface.
- d. *Transfer Protocol.* A value of 0x10 is written to the A/D control register, H11ADCTL. When bit 7 of the register goes high (1) register H11ADR3 is read to determine the voltage level of the HHM batteries.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.4 Digital RF Transceiver Interfaces.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The transmit data interface (IF DRTA) is capable of sending two messages. The formats of the transmission of the PMC positive acknowledgment (PPA) and PMC soldier status (PSS) are shown in tables 4 and 5.

The receive data interface (IF DRTB) can receive one message. The format for the request for soldier status (CSS) message is given in table 3.

- c. *Memory Buffer And Location.* The memory buffer necessary for the digital RF transceiver consists of two 200-byte output buffers, "tpakbuf" and "pkmess," and one 10 x 50 byte input buffer, "pkbuf."
- d. *Transfer Protocol.* Serial communication is used in both the transmit and receive data protocols. To transmit a message, the serial communication status register, H11SCSR, is polled until the transmitter-ready bit goes high (1). The character to be transmitted is then written to the serial communication data register, H11SCDR. To receive a message, the serial communication receiver interrupts the processor when a character is received. The interrupt routine implements a finite state machine to recognize valid packets.
- e. *Initiation Condition.* Table 2 indicates general conditions for the initiation of all three signals pertaining to the digital RF transceiver interface. The DRTA interface can send two messages, the PPA message and the PSS message, which the digital RF transceiver will broadcast to the HHM. The PPA message verifies the PMC has been activated and the PSS message transmits the status information. The DRTB interface receives one message, the CSS message. This message requests the PMC to transmit status information.
- f. *Priority Level.* The receive data has the highest priority.
- g. *Expected Response.* Table 2 shows the expected responses for all three signals. The receive data interface (DRTB IF) is the only interface with timing restrictions. A character received at the DRTB interface causes an interrupt to be sent to CSCI.1 when the character is being processed. The maximum response time per character is approximately 1/30 second which is the time necessary for character transmission at 300 baud. If incoming characters are not responded to in time, then the receive character buffer will be overrun causing the character to be lost. The message will need to be retransmitted. This is not fatal unless characters are lost in every packet.

3.3.3.2.5 Real Time Clock Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The signal is an interrupt that is emitted every 20 milliseconds.
- c. *Memory Buffer And Location.* This section is not applicable to this interface.
- d. *Transfer Protocol.* This section is not applicable to this interface.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.

- g. *Expected Response.* Table 2 shows the signal causes an internal interrupt at 50 Hertz. The interrupts generated are necessary for the A/D conversions, the battery voltage checks, and for functions with elapsed time counters and countdown timers.

3.4 *Detailed Functional and Performance Requirements.*

3.4.1 *Power Up Initialization Function.*

3.4.1.1 *Inputs.* The function has 1 input which is the power up reset. The reset serves as a hardware and software initialization. See the table for the power up function input (Table 6) for additional information.

3.4.1.2 *Processing.*

- a. *Intent.* This function initializes the 68HC11 microcomputer hardware, the interrupt vectors, and the program variables.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The eight independent initializations in this function are described as follows. All program variables are assigned values of zero. The down counter is assigned the value stored in the constant "EVALTIME." The routine "key" turns off the key line to the transmitter as soon as possible since the initial hardware power-up causes the transmitter to be keyed. The routine "pkclear" initializes the packet-receiving finite state machine. Routine "int_setup" initializes the interrupt vectors for the interrupt routines OC1INT, INTINT, COPINT, and SCIINT. The routine also initializes the output compare to 1, and enables the timer interrupt. Routine "tty_setup" sets the serial interface for one start bit and eight data bits ($H11SCCR1 = 0$), enables the transmitter, receiver, and receiver interrupt ($H11SCCR2 = 2C$), and sets the baud register for 300 baud ($H11BAUD = 0x35$). The routine "adon" turns on the A/D converter. The routine "stop_enable" enables the HC6811 STOP (low power standby mode) instruction.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.1.3 *Outputs.* This function has 23 initialized outputs. The buffer and stack indexes "pin," "pout," "S1," "S2," and "S3" are cleared. The flag "adflag" is set to 1 to show the

A/D converter is enabled and the flag "adnow" is set to 0 to indicate an A/D conversion is not in progress. The flag "daemonflag" is cleared to show no packets are being processed. The flags "evflag," "proxflag," and "req s" are all set to 0 to indicate there have been no previous evaluation requests, PRCV interrupts, or soldier status requests, respectively. The two timers, "dcount" and "pcount," are cleared and the down counter, "dsec," is assigned the value of the constant "EVALTIME." An initialization subroutine (called by the function) clears all storage necessary for packet construction. The remaining 10 outputs are hardware oriented. The key line, H11PORTA (bit 4), is set inactive. Both the A/D converter, H11OPTION, and the wrist impedance transmit, H11PORTA (bit 5), are powered up. Register H11PORTD is cleared. The register H11BAUD is set for a rate of 300 baud. The port D direction register, H11DDRD, is initialized to 2. Both the serial interface register, H11SCCR1, and the output compare register, H11TOC1, are cleared. And the serial interface interrupt, H11SCCR2, and the output compare interrupt, H11TMSK1, are enabled. See the table for the power up function output (Table 7) for additional information.

3.4.2 Main Monitor Function.

3.4.2.1 *Inputs.* The function has 5 inputs which all deal with evaluation requests. The flag "evflag" signals requests for soldier evaluation. The variable "reqps" records HHM requests for soldier status that have been postponed due to an A/D conversion. The variables "hrate" and "mocount" store soldier evaluation results. The down-count timer "dsec" is used to sense inactivity in the communication channel. See the table for the main monitor function input (Table 8) for additional information.

3.4.2.2 Processing.

- a. *Intent.* This function has three purposes. First, the function monitors the incoming packets for evaluation requests. The function answers the requests by triggering the evaluation function and then responding via the transceiver to the querying HHM. Secondly, when the function detects periods of inactivity exceeding a specified time, the function switches the hardware to low power standby mode (STOP on 68HC11). The function also detects proximity receiver interrupts and returns the hardware to active mode.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The function is a continuous loop which checks for evaluation requests. When a request is made, the analog data from the wrist impedance are converted and processed. If the down counter (dsec) reaches a value of zero, the A/D converter and the radio are turned off causing the PMC to "sleep." A proximity interrupt turns the A/D converter on and resets the down counter so evaluation can begin again. See the flowchart for the main monitor function (Figure 3) for additional detail.

- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.2.3 *Outputs.* This function has 4 outputs. The variables "hrate" and "mocount" contain the heart rate and the relative motion value, respectively. The variable "evnum" contains the number of evaluations performed since the system was powered-up. The counter "dsec" is the down counter necessary to detect inactivity in the communication channel. See the table for the main monitor function output (Table 9) for additional information.

3.4.3 *Proximity Interrupt Function.*

3.4.3.1 *Inputs.* The function has 2 inputs which relate to the proximity receiver. The PRCV interrupt activates the PMC. The flag "proxflag" indicates whether the interrupt has occurred. See the table for the proximity interrupt function input (Table 10) for additional information.

3.4.3.2 *Processing.*

- a. *Intent.* This function responds to HLM requests for activation through the proximity receiver hardware and the maskable interrupt (IRQ bar). The function is capable of interrupting the 68HC11 STOP (low power standby mode) state when the low power mode is being maintained.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The following sequence of steps is performed by the interrupt. Initially the flags "proxflag" and "stompflag" are set. Then the interrupts are enabled. The radio is turned on and an acknowledge packet (PPA) is transmitted. The flags "evflag" and "didtrans" are set to 1 before the interrupts are disabled. Then the flag "proxflag" is set to 0 and the interrupts are enabled again.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.3.3 Outputs. The function has 5 outputs. The flag "evflag" signals requests for soldier evaluation. The flag "didtrans" indicates the RF transmission has been activated during the A/D conversion. And the flag "stompflag" serves as the timer flag. The H11SCCR2 register is set to enable the serial interface interrupt and the H11PORTA register (bit 4) is set to inactivate the radio power key line. See the table for the proximity interrupt function output (Table 11) for additional information.

3.4.4 Clock Interrupt Function.

3.4.4.1 Time Keeping Function.

3.4.4.1.1 Inputs. The function has one interrupt and three counters as inputs. The clock interrupt provides for accurate time keeping. The counter "dsec" is the down counter used to detect inactivity in the communication channel. The counters "dcount" and "pcount" are general purpose timers used for delays. See the table for the time keeping function input (Table 12) for additional information.

3.4.4.1.2 Processing.

- a. *Intent.* The function provides a regular interrupt at a frequency of 50 Hz for various time keeping functions.
- b. *Parameters Affected.* The output compare register of the 68HC11 is continuously updated to permit an interrupt to occur every 20 milliseconds.
- c. *Sequence and Timing.* Initially, the first timer interrupt flag register is cleared by writing 0x80 to the H11TFLG1 register. Then the H11TOC1 register is incremented by 0x9c40 to create an interrupt at the end of 20 milliseconds. The computer operating properly timeout interrupts are prevented by assigning the values 0x55 and 0xaa to the H11COPRST register. The counter variable "clk50" is incremented. If the variable reaches a value of 50 then the counter is reset. The down counters "dcount" and "pcount" are decremented if they have values greater than zero. The down counter variable "dsec" is decremented at one second intervals as long as the variable is greater than zero.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.4.1.3 Outputs. The function has 6 outputs. The down-count timer "dsec" is used to

sense inactivity in the communication channel. The variables "dcount" and "pcount" are general timers used for delays. The register H11TFLG1 is set so the interrupt flag is cleared. The register H11TOC1 is loaded to provide output comparison interrupts at a rate of 50 Hz and H11COPRST is set to postpone COP timer interrupts. See the table for the time keeping function output (Table 13) for additional information.

3.4.4.2 A/D Conversion Function.

3.4.4.2.1 *Inputs.* The function requires 5 inputs to perform the A/D conversion. The flag which enables the A/D conversions is "adflag." The A/D conversion status byte is stored in the register H11ADCTL. And the registers H11ADR1, H11ADR2, and H11ADR3 contain the wrist impedance signal, the off-wrist signal and the battery voltage, respectively. See the table for the A/D conversion function input (Table 14) for additional information.

3.4.4.2.2 Processing.

- a. *Intent.* This function performs the A/D conversion of the data from the wrist impedance circuit.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The necessary A/D conversion rate is 25 Hz. Since the internal interrupt occurs at a rate of 50 Hz, the conversion is triggered by odd interrupts (variable "clk50" is odd). Conversion is only possible when "adflag" indicates data are present. The conversion begins by setting the H11ADCTL register of the 68HC11 to 0x10 which sets bit 4 of this register set to 1. A conversion is complete when bit 7 of the register goes high. Following the conversion, the A/D values are stored in the variables "adval," "ad_offw" and "ad_batt" and the flag "adnext" is set high to signal conversion completion.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.4.2.3 *Outputs.* The function has 5 outputs. The variable "adval" contains the wrist impedance signal from the A/D conversion function. The flag "adnext" signals completed A/D conversions. The variables "ad_offw" and "ad_batt" contain the off-wrist signal and the battery voltage, respectively. The register H11ADCTL activates the A/D converter. See the table for the A/D conversion function output (Table 15) for additional information.

3.4.5 *Serial Receive Interrupt Function.*

3.4.5.1 *Packet Construction Function.*

3.4.5.1.1 *Inputs.* The function has 9 inputs. The serial receiver interrupt indicates the signal reception of a character by the digital RF transceiver (DRT). The variables "pin" and "pout" are input and output index pointers used in storing packet information. The variable "pcount" is a general timer used for delays. The flag "daemonflag" is used to insure only one invocation of the input packet processing will run at a time. The flag "stompflag" detects the transmission of a PPA packet while a PSS packet is being sent. The initializing subroutine "pkclear" clears the buffer area necessary for packet construction. The register H11SCSR stores the status byte from the serial receiver. And the register H11SCDR contains the received character. See the table for the packet construction function input (Table 16) for additional information.

3.4.5.1.2 *Processing.*

- a. *Intent.* This function accepts input characters from the serial interface and attempts to construct a legal packet from the characters.
- b. *Parameters Affected* No parameters are affected by this function.
- c. *Sequence and Timing.* Packet construction is accomplished by a finite state machine. Figure 4 shows the necessary flow with the variable "rstate" holding the current state of the machine.
- d. *Error Detection and Recovery.* An error routine checks the validity of the transmitted packet. The routine sums the characters in the packet to get an 8 bit checksum. Then the last two characters in the packet, which are a hex ASCII representation of the checksum, are decoded. The two checksums are compared and if they are not equal then an error occurred during packet transmission. An error results in the packet being discarded and a new packet being constructed.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.5.1.3 *Outputs.* This function has 3 outputs. The array "pkbuf" contains packet information. The variables "pin" and "pout" are the array indexes which point to the current input and output packet locations in "pkbuf." See the table for the packet construction function output (Table 17) for additional information.

3.4.5.2 *Input Packet Processing Function.*

3.4.5.2.1 Inputs. This function requires 5 inputs to process incoming packets. The array "pkbuf" is a buffer for packet storage. The variables "pin" and "pout" are the array indexes which point to the input and output packet locations in "pkbuf." The flag "adnow" indicates when an A/D conversion is in progress. The function uses the down counter "dsec." See the table for the input packet processing function input (Table 18) for additional information.

3.4.5.2.2 Processing.

- a. *Intent.* This function decodes valid packets to determine if the receiving PMC is the correct destination and then responds appropriately.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The packet address is first decoded. If the address matches the PMC address then the processing continues with checks to assure the request is for soldier status (CSS) and the packet can be processed. Proper requests cause a PMC soldier status (PSS) packet to be transmitted. Requests made during A/D conversions must be postponed. These requests are recorded by the variable "reqpss" and no PSS packets are transmitted. See the flowchart for the input packet processing (Figure 5) for additional detail.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.5.2.3 Outputs. The function has 2 outputs. The flag "daemonflag" assures only one invocation of the input packet processing is running at a time. The variable "reqpss" records IHM requests for soldier status that have been postponed due to an A/D conversion. See the table for the input packet processing function output (Table 19) for additional information.

3.4.6 Get Data Function.

3.4.6.1 Inputs. This function has 3 inputs. The variable "adval" contains the wrist impedance signal from the A/D conversion function. The two other inputs are flags. The flag "adnext" signals completed A/D conversions. And the flag "didtrans" indicates the RF transmission has been activated during the A/D conversion. See the table for the get data function input (Table 20) for additional information.

3.4.6.2 Processing.

- a. *Intent.* This function receives data from the wrist impedance circuit and stores the A/D converted data points in an array.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The following sequence of events occurs for each data point stored in the array "pbuf." The flag "adnext" is cleared. The clock interrupt routines (functions Time Keeping and A/D conversion) execute an A/D conversion and save the converted value in the variable "adval." The flag "adnext" is then set to 1 to indicate the conversion is complete and the data point has been processed. Then the value in "adval" is transferred to the next location in "pbuf." This entire sequence is repeated until "pbuf" has been filled.
- d. *Error Detection and Recovery.* Another function may transmit data while "pbuf" is being filled. This will cause the data in "pbuf" to be contaminated. When this contamination occurs the flag "didtrans" is set to 1, the data in "pbuf" is discarded, and filling begins again.
- e. *Restrictions or Limitations.* A transmission error will occur if an A/D data point is moved to the storage array "pbuf" while any other signal is being sent.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.6.3 *Outputs.* The function has 3 outputs. The array "pbuf" contains the digitized impedance signal and the variable "npbuf" holds the length of this array. The flag "adnow" indicates when an A/D conversion is in progress. See the table for the get data function output (Table 21) for additional information.

3.4.7 Get Rates.

3.4.7.1 Preprocessor Function.

3.4.7.1.1 *Inputs.* The function has 2 inputs. The array "pbuf" stores the digitized impedance signal. The variable "npbuf" indicates the number of array storage locations currently in use. See the table for the preprocessor function input (Table 22) for additional information.

3.4.7.1.2 Processing.

- a. *Intent.* This function locates and filters A/D data. The function also determines the length of the filtered segment.
- b. *Parameters Affected.* No parameters are affected by this function.

- c. *Sequence and Timing.* The function locates all on-scale data points. From these values, the longest continuous on-scale data section is isolated. This section is initially passed through a low pass filter and then the data segment is passed through a high pass filter. The average of the resulting data segment is determined. Based on the median value, a new on-scale upper limit is computed. The current data segment is passed through the newly established window and again the longest continuous data section is isolated. The length of the data segment is determined and stored. See flowchart for the preprocessor function (Figure 6) for additional information.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.7.1.3 *Outputs.* The function has 4 outputs. The array "pbuf" contains the filtered A/D data segment. And the variable "seglen" holds the length of this data segment. The average value of the data segment is stored in the variable "val." The variable "mocount" contains the motion count. See the table for the preprocessor function output (Table 23) for additional information.

3.4.7.2 *Heart Rate Calculation Function.*

3.4.7.2.1 *Inputs.* This function has 5 inputs. The array "pbuf" contains the filtered A/D data segment. The length of this data segment is stored in the variable "seglen." The stack indexes ("S1", "S2", "S3") locate values for signal averaging. And the variable "val" holds the data segment average. The motion count is stored in the variable "mocount." See the table for the heart rate calculation function input (Table 24) for additional information.

3.4.7.2.2 *Processing.*

- a. *Intent.* This function calculates the heart rate of a filtered data segment with a minimum duration of five seconds.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* Initial checks are made to determine if the data are reasonable and if the data have a duration of at least five seconds. If processing is to continue, the following steps are performed. The FFT of the data segment is computed and the spectrum ($S(f)$) is calculated. Depending upon the number of consecutive data segments previously processed, the current

spectrum is averaged with zero, one or two previous spectra. The frequency corresponding to the peak amplitude of the average spectra ($S(i)$) is located and the heart rate is computed using the formula

$$\text{HeartRate} = \frac{(i-2)*S(i-2) + i*S(i-1) + (i)*S(i)}{S(i-2) + S(i-1) + S(i)} * \frac{750}{128} \text{ bpm} .$$

See the flowchart for the heart rate function (Figure 7) for additional information.

- d. *Error Detection and Recovery.* Periodogram averaging reduces the effects of the noise in the signal.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.7.2.3 Outputs. This function has only one output which is the variable "hrate." This variable stores the soldier's heart rate. See the table for the heart rate calculation function output (Table 25) and the table for heart rate codes (Table 28) for additional information.

3.4.8 Transmit Function.

3.4.8.1 Inputs. This function has 7 inputs. The variable "evnum" contains the number of evaluations performed since power-up. The variables "ad_offw," "hrate," "mocount," and "ad_batt" contain the off-wrist signal, the heart rate, the relative motion value, and the battery voltage, respectively. The function uses the general timer "dcount." The register H11SCSR stores the status byte from the serial receiver. See the table for the transmit function input (Table 26) for additional information.

3.4.8.2 Processing.

- a. *Intent.* This function encodes and transmits valid packets to requestings H1Ms.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The sequence necessary to build and transmit a valid packet is as follows. The packet characters are encoded and stored in the buffer "pkmess." The routine "buildpak" adds the packet header ("A"), the PMCID, and the id end marker (:) to the existing packet characters. The entire string is stored in the buffer "tpakbuf." Then the routine "sendpak" calculates and encodes the packet's 8 bit checksum which is added to the packet along with a line feed and carriage return. Once the packet is built the transmitter is keyed. Following a period of 200 msec to allow the HHM

receiver to stabilize, the packet is sent. The transmit key line is then turned off.

- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* The transmitter must be keyed for 200 msec before data are sent to give the modem in the receiver the time necessary to decode valid data.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.8.3 Outputs. This function has 5 outputs. The flag "evflag" signals requests for soldier evaluation. The flag "didtrans" indicates the RF transmission has been activated during the A/D conversion. The array "tpakbuf" contains processed information to be displayed. The register H11SCDR contains the received characters and the register H11PORTA (bit 4) controls the radio power key line. See the table for the transmit function output (Table 27) for additional information.

3.5 *Adaptation Requirements.*

3.5.1 System Environment. This section is not applicable to this specification.

3.5.2 System Parameters. This section is not applicable to this specification.

3.5.3 System Capacities. This section is not applicable to this specification.

3.6 *Quality Factors.*

3.6.1 Correctness Requirements. The PMC CSCI is considered to be 100% correct.

3.6.2 Reliability Requirements. The software has been developed to consistently provide meaningful results regardless of the input signal.

3.6.3 Efficiency Requirements. The CSCI uses the computer resources in a manner which accomplishes status cycle evaluation and reporting in less than 15 seconds.

3.6.4 Integrity Requirements. No control against unauthorized access to operations and data exists in this CSCI.

3.6.5 Usability Requirements. This section is not applicable to this specification since there is no direct interaction between the user and this CSCI.

3.6.6 *Maintainability Requirements.* The maximum effort necessary to locate and fix an error should not exceed 2 human work weeks.

3.6.7 *Testability Requirements.* Testing of the CSCI should range from 1 day to 1 week depending on the data set selected. Sections of the CSCI are difficult to independently test due to their close relationship to the hardware.

3.6.8 *Flexibility Requirements.* Minimal time will be required for enhancements due to the modular nature of the PMC CSCI.

3.6.9 *Portability Requirements.* The software is portable due to the facts it is primarily written in C and it is modular. However, since the software has been developed for a particular microprocessor with built-in interfaces, complications may result when attempting to re-port the software.

3.6.10 *Reusability Requirements.* The PMC CSCI will be reusable in other applications with minimal effort since the software is written in the high level language C and is modular.

3.6.11 *Interoperability Requirements.* The CSCI is not compatible with any existing systems except for the hand held monitor (HHM) and the prototype multimonitor.

3.6.12 *Additional Quality Factor Requirements.* This section is not applicable to this specification.

3.7 *CSCI Support.*

3.7.1 *Facility Requirements.* Laboratory workspace suitable for the equipment and personnel listed below is necessary. The workspace will need to have standard 120 volt AC power, temperature control for the room, a computer table, and a workbench.

3.7.2 *Equipment Requirements.* The following equipment is necessary:

- a. IBM PC compatible computer with a serial port to talk to the emulator (recommend a 386 system with a hard disk drive)
- b. Motorola HDS-300 Development System with a 68HC11 emulator pod
- c. Wire-wrap prototype of PMC and HHM hardware with a digital to analog converter for debugging and testing software during development
- d. 100 MHz bandwidth oscilloscope to monitor software functions
- e. Five volt DC power supply for prototype circuit

- f. PROM programmer to attach to the IBM PC compatible computer for programming 27C64 and 27C256 EPROMS

3.7.3 *Software Requirements.* The following software is necessary:

- a. INTRC-C/68HC11 C cross compiler for the Motorola 68HC11 microprocessor which runs on the IBM PC compatible computer
- b. MS-DOS for the IBM PC compatible computer
- c. Text editor
- d. Copies of the PMC and HHM source code
- e. File transfer program for PC to HDS-300 communication
- f. Digital signal processing software consisting of a database of wrist impedance signals and appropriate analysis and display software
- g. C compiler for the IBM PC compatible computer to compile and test the software described in item 6 (above)

3.7.4 *Personnel Requirements.* The personnel maintaining and developing the PMC and HHM software need knowledge in the following areas:

- a. C programming
- b. stand-alone microprocessor programming (MC68HC11 in particular)
- c. real time programming (interrupt handling, etc.)
- d. digital signal processing

3.8 *Traceability.* This section is not applicable to this specification.

4. QUALIFICATION REQUIREMENTS

4.1 *General Qualification Requirements.* Testing was performed on the heart rate processing software of the PMC. See Appendix I for details of the testing. Additional testing of the CSCI was done in conjunction with the HHM CSCI (HHM-CSCI.2) at the interface level. See documentation concerning this testing in the Interface Requirement Specification (A006).

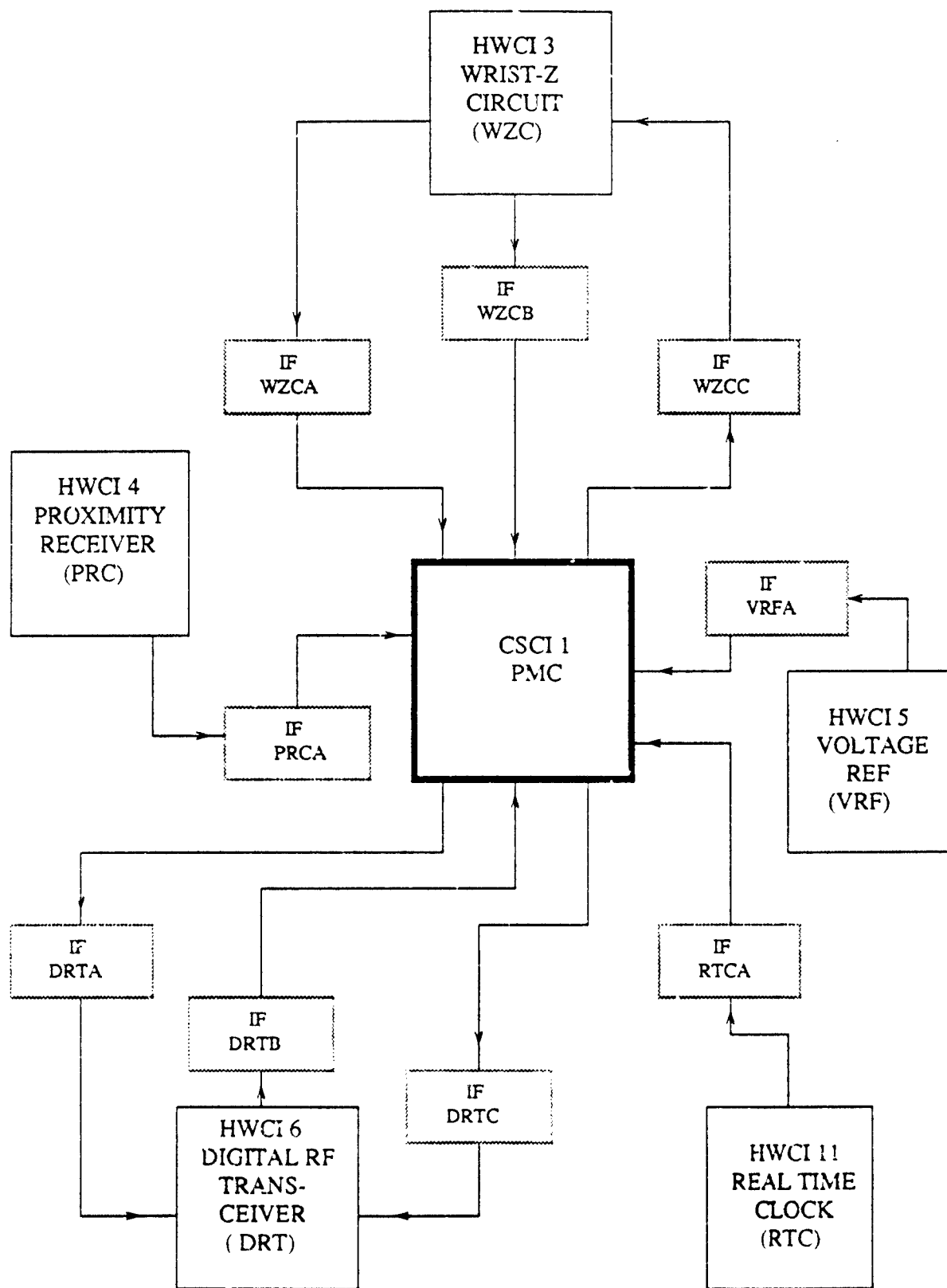
4.2 *Special Qualification Requirements.* This section is not applicable to this specification.

5. PREPARATION FOR DELIVERY

5.1 *Preparation For Delivery* The CSCI software will be delivered in object code form on an EPROM soldered into the PMC unit. A hardcopy of the source code can be found in the appendix which accompanies this documentation.

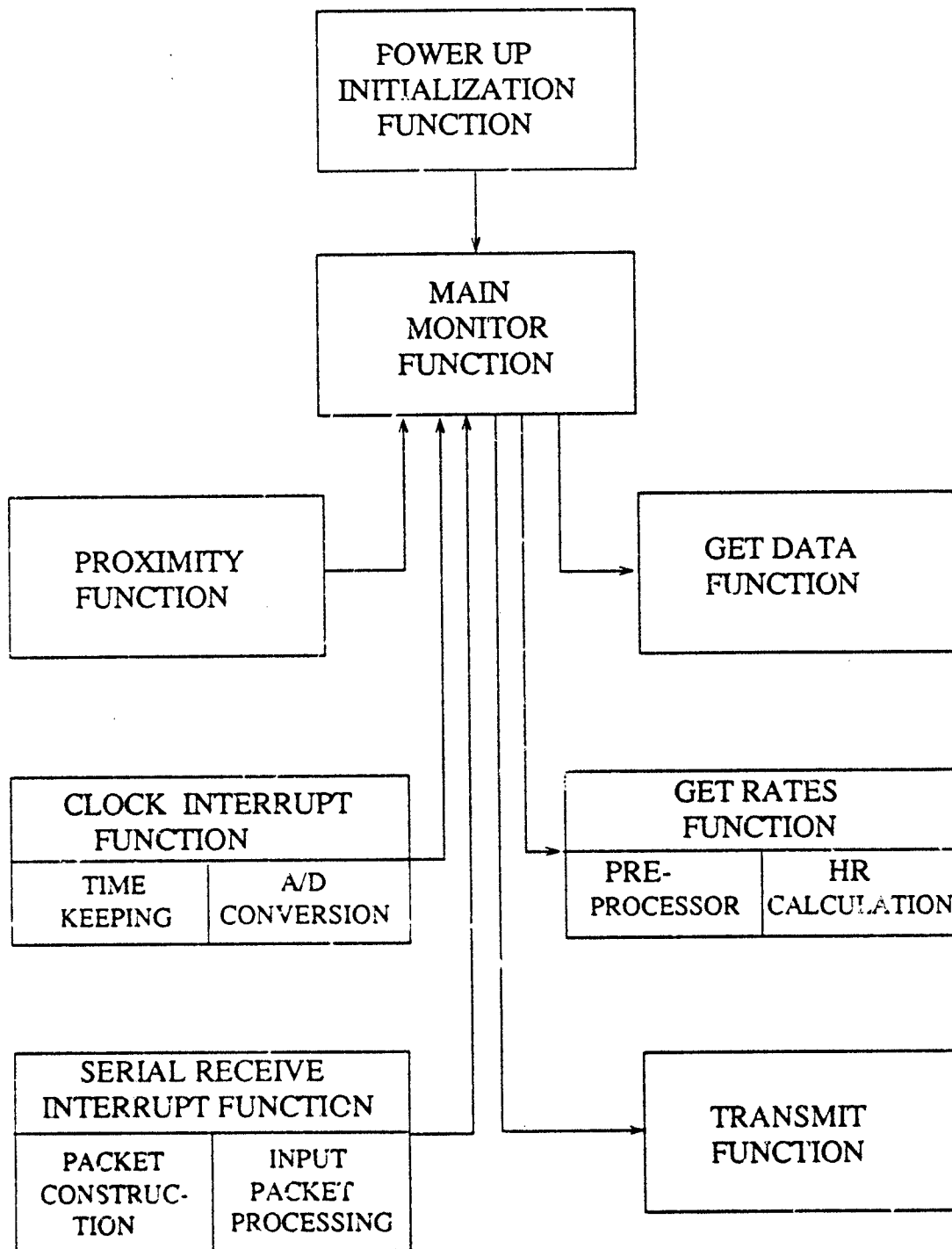
6. NOTES

6.1 *Acronyms* See acronym summary table (Table 29).



NOTE:
IF = INTERFACE

FIGURE 1. Interface block diagram.

FIGURE 2. *Function block diagram.*

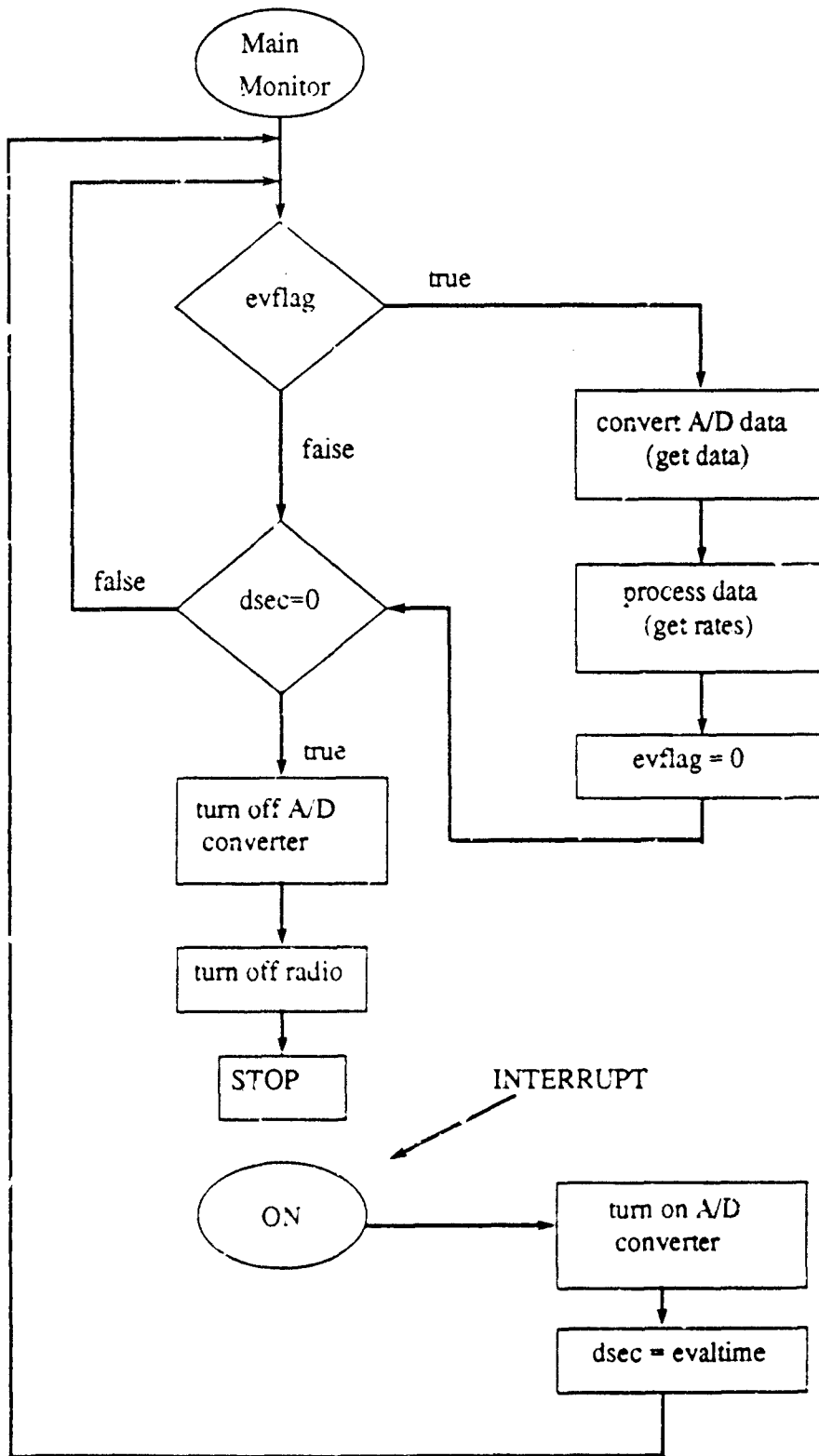


FIGURE 3. Funtion main monitor flowchart.

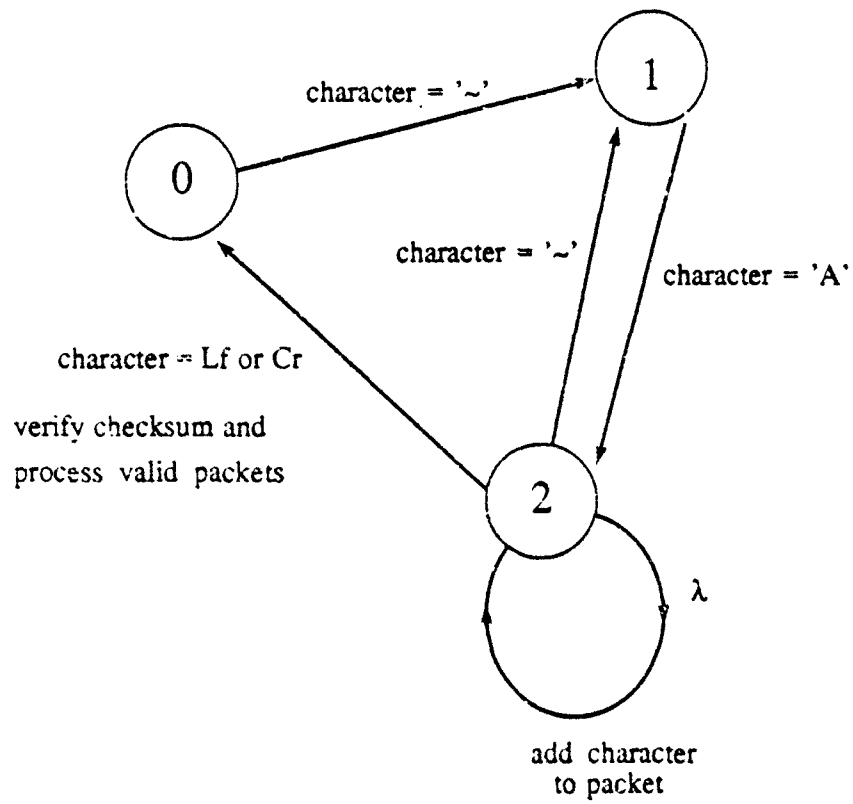


FIGURE 4. *Function packet construction finite state machine.*

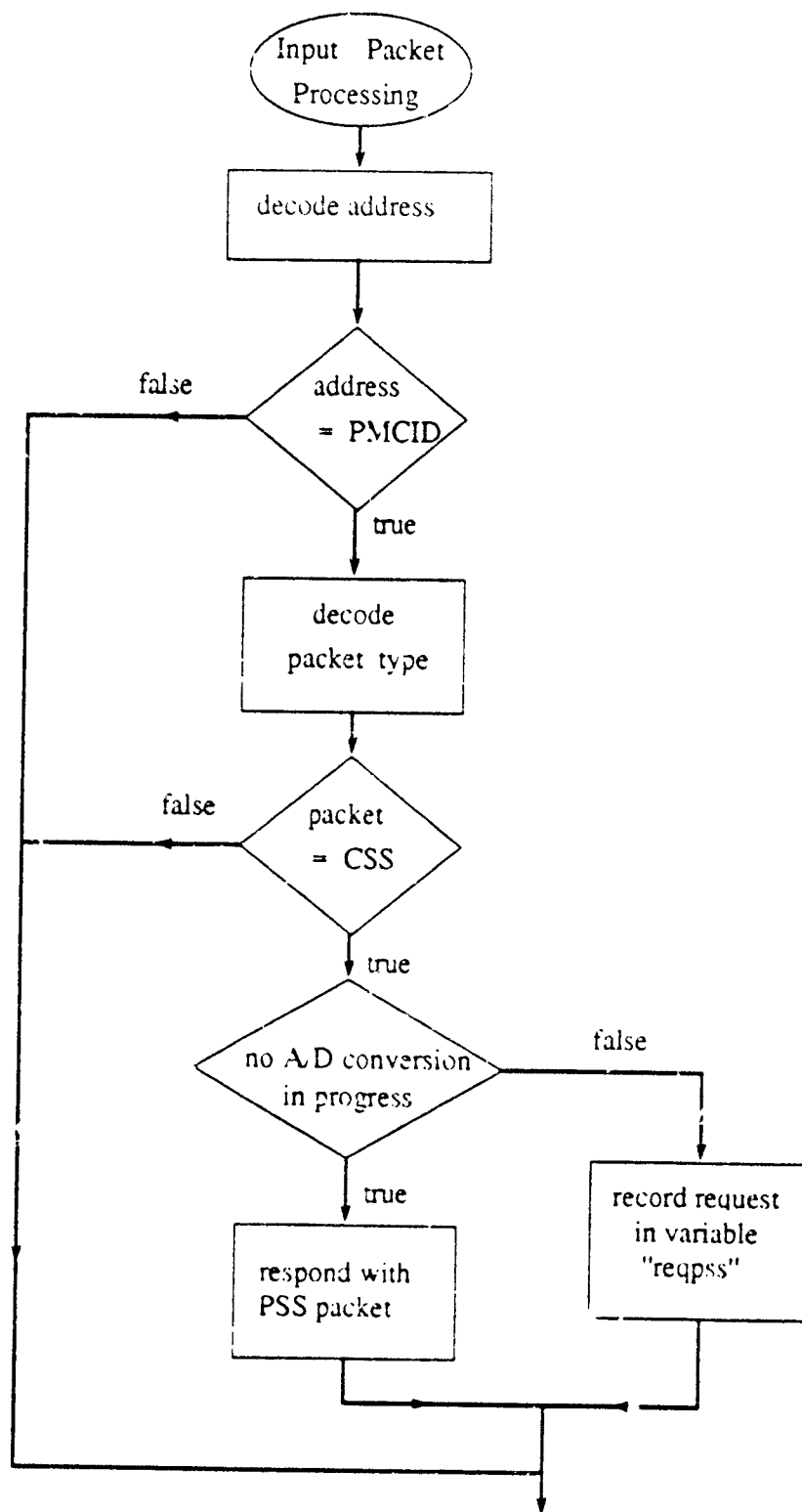


FIGURE 5. Function input packet processing flowchart.

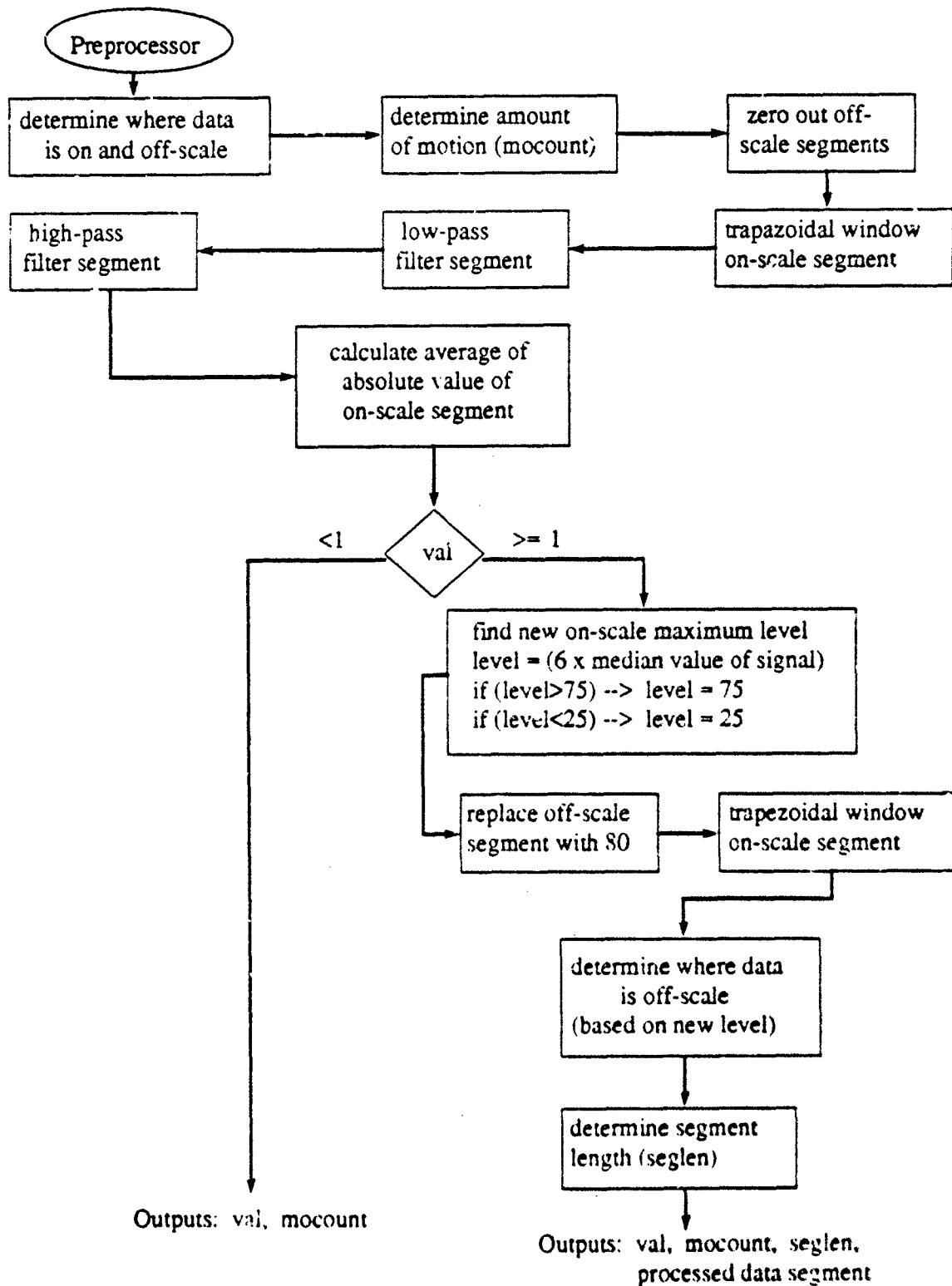


FIGURE 6. Function preprocessor flowchart.

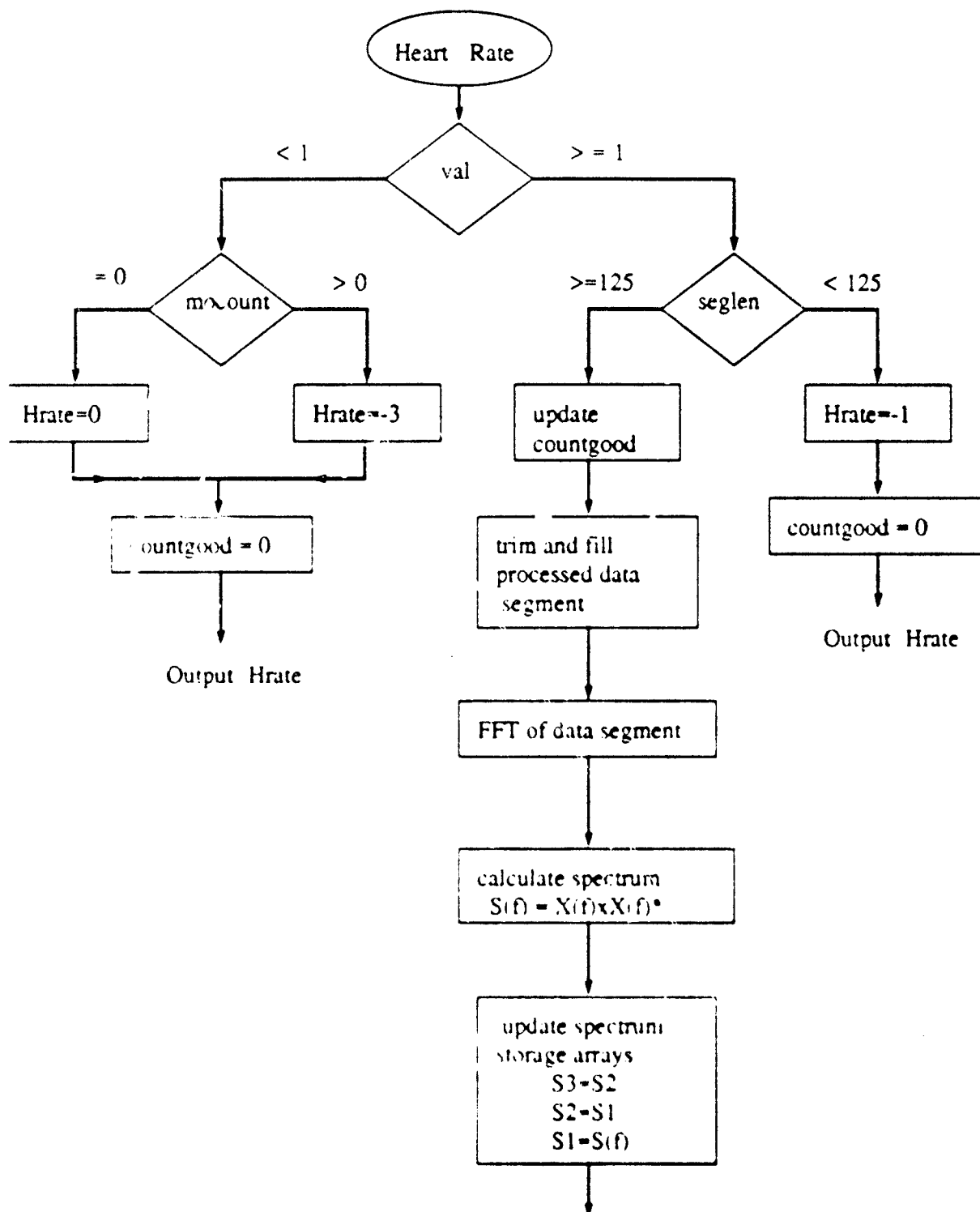


FIGURE 7. Function heart rate flowchart.

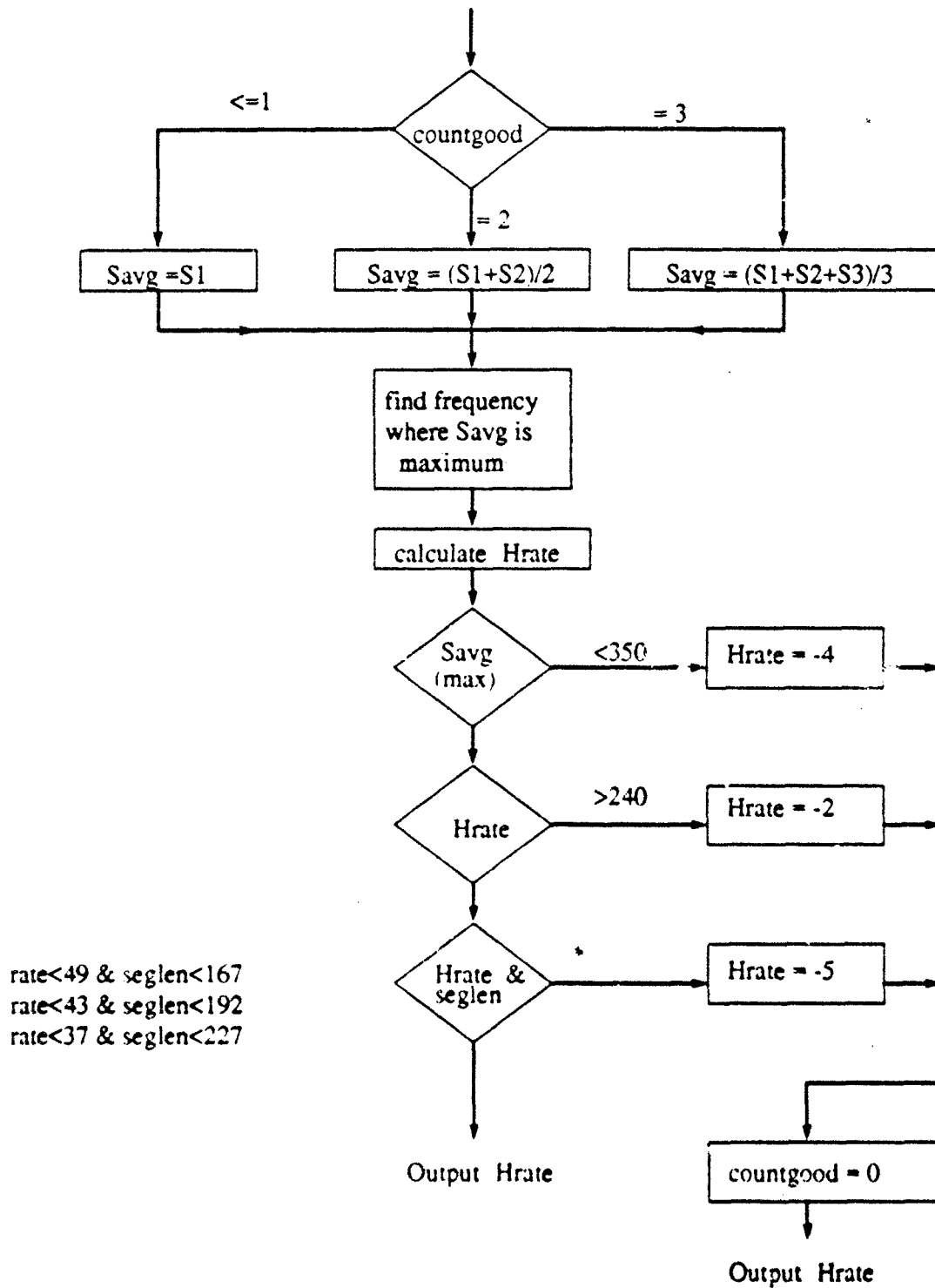


FIGURE 7. continued
Function heart rate flowchart.

TABLE 1. Interface identification/documentation.

INTERFACE NAME	INTERFACING ITEM		INTERFACE DOCUMENT NAME	INTERFACE DOCUMENT PARAGRAPH
	TITLE	CI NUMBER		
WZCA	WRIST-Z CIRCUIT	3	SRS for PMC -- WZC interface	3.3.3.2.1
WZCB	WRIST-Z CIRCUIT	3	SRS for PMC -- WZC interface	3.3.3.2.1
WZCC	WRIST-Z CIRCUIT	3	SRS for PMC -- WZC interface	3.3.3.2.1
PRCV	PROXIMITY RECEIVER	4	SRS for PMC -- PRCV interface	3.3.3.2.2
VRFA	VOLTAGE REF	5	SRS for PMC -- VRFA interface	3.3.3.2.3
DRTA	DIGITAL RF TRANSCEIVER	6	SRS for PMC -- DRT interface	3.3.3.2.4
DRTB	DIGITAL RF TRANSCEIVER	6	SRS for PMC -- DRT interface	3.3.3.2.4
DRTC	DIGITAL RF TRANSCEIVER	6	SRS for PMC -- DRT interface	3.3.3.2.4
RTCA	REAL TIME CLOCK	11	SRS for PMC -- RTCA interface	3.3.3.2.5

TABLE 2. *Interface summary.*

INTERFACE NAME	INFORMATION DESCRIPTION	INITIATION CONDITION	EXPECTED RESPONSE
WZCA	Wrist impedance analysis signal HWCI 3-to-CSCI 1	WZC powered up	Signal that varies with contraction of heart
WZCB	Off-wrist analog signal HWCI 3-to-CSCI 1	WZC powered up	Signal that varies with electrode contact
WZCC	WZC power CSCI 1-to-HWCI 3	PMC activated	Activates WZCA and WZCB
PRCV	Proximity data HWCI 4-to-CSCI 1	Activated HHM nearby	Interrupt sent to activate PMC
VRFA	Analog reference voltage HWCI 5-to-CSCI 1	PMC activated	Voltage reference
DRTA	Transmit data serial communication CSCI 1-to-HWCI 6	PMC sending data packet	Data sent by RF transceiver
DRTB	Receive data serial communication HWCI 6-to-CSCI 1	DRT receiving data packet	Data packet processed
DRTC	DRT power CSCI 1-to-HWCI 6	PMC activated	DRT activated
RTCA	Real time clock interrupt HWCI 11-to-CSCI 1	HHM powered up or PMC activated	Internal interrupt at 50 HZ

TABLE 3. *Request for soldier status (CSS) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	CSS	packet type
1 byte*	ASCII		space (blank)
1 byte*	ASCII	I	signals CSCI 1 response
2 bytes	ASCII	hex digits	packet checksum
2 bytes	ASCII	LfCr	packet trailer

* optional

TABLE 4. *PMC positive acknowledge (PPA) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	PPA	packet type
2 bytes	ASCII	hex digits	packet checksum
2 bytes	ASCII	LfCr	packet trailer

TABLE 5. *PMC soldier status (PSS) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	PSS	packet type
1 byte	ASCII		space (blank)
1 to 5 bytes	ASCII	1 to 5 digits	evaluation count
1 byte	ASCII		space (blank)
1 to 3 bytes	ASCII	1 to 3 digits	off-wrist level
1 byte	ASCII		space (blank)
2 to 3 bytes	ASCII	2 to 3 digits	heart rate
1 byte	ASCII		space (blank)
1 to 4 bytes	ASCII	1 to 4 digits	motion count
1 byte	ASCII		space (blank)
1 to 3 bytes	ASCII	1 to 3 digits	A/D reference level
1 byte	ASCII		space (blank)
2 bytes	ASCII	LfCr	packet trailer

TABLE 6. *Function power up initialization input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Power up reset	N/A	Once when device is turned on	N/A	CSCI 1

TABLE 7. *Function power up initialization output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Buffer input index (pin)	N/A	N/A	N/A	Function Packet Construction
2	Buffer output index (pout)	N/A	N/A	N/A	Function Packet Construction
3	Stack indexes (S1, S2, S3)	N/A	N/A	N/A	Function Heart Rate Calculation
4	A/D conversion enable flag (adflag)	N/A	N/A	N/A	Function A/D Conversion
5	Conversion in progress flag (adnow)	N/A	N/A	N/A	Function Input Packet Processing
6	Individual processing flag (daemonflag)	N/A	N/A	N/A	Function Packet Construction
7	Evaluation flag (evflag)	N/A	N/A	N/A	Function Main Monitor
8	PRCV interrupt flag (proxflag)	N/A	N/A	N/A	Function Proximity Interrupt

TABLE 7, continued
Function power up initialization output

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
9	Record flag (reqpss)	N/A	N/A	N/A	Function Main Monitor
10	Down counter (dsec)	Seconds	N/A	Yes	Function Main Monitor or Time Keeping or Input Packet Processing
11	General timer (dcount)	Seconds	N/A	Yes	Function Time Keeping
12	General timer (pcount)	1/50 Seconds	N/A	Yes	Function Power Up
13	Initialization subroutine (pkclear)	N/A	N/A	N/A	Function Packet Construction
14	Baud rate register (H11BAUD)	N/A	N/A	N/A	HWCI 6
15	Port D direction register (H11DDRD)	N/A	N/A	N/A	HWCI 6

TABLE 7, continued
Function power up initialization output

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
16	A/D conversion (H11OPTION)	N/A	N/A	N/A	HWCI 3
17	Key line (H11PORTA bit 4)	N/A	N/A	N/A	HWCI 6
18	Wrist impedance (H11PORTA bit 5)	N/A	N/A	N/A	HWCI 3
19	Port D (H11PORTD)	N/A	N/A	N/A	HWCI 6
20	Serial interface register 1 (H11SCCR1)	N/A	N/A	N/A	HWCI 6
21	Serial interface interrupt (H11SCCR2)	N/A	N/A	N/A	HWCI 6
22	Output compare 1 interrupt (H11TMSK1)	N/A	N/A	N/A	HWCI 11
23	Output compare 1 register (H11TOC1)	N/A	N/A	N/A	HWCI 11

TABLE 8. *Function main monitor input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Evaluation flag (evflag)	N/A	15 second intervals when PMC activated	N/A	Function Power Up or Proximity Interrupt or Transmit
2	Record flag (reqpss)	N/A	N/A	N/A	Function Power Up or Input Packet Processing
3	Heart rate (hrate)	Beats/Min	15 second intervals when PMC activated	No	Function Heart Rate Calculation
4	Motion count (mocount)	N/A	15 second intervals when PMC activated	No	Function Preprocessor
5	Down counter (dsec)	Seconds	1/second	No	HWCI 11 or Function Power Up or Time Keeping

TABLE 9. *Function main monitor output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Heart rate (hrate)	Beats/Minute	N/A	No	Function Transmit
2	Relative motion (mocount)	N/A	N/A	No	Function Transmit
3	Evaluation number (evnum)	N/A	N/A	No	Function Transmit
4	Down counter (dsec)	Seconds	N/A	No	Function Time Keeping

TABLE 10. *Function proximity interrupt input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	PRCV interrupt	N/A	Once to activate PMC	N/A	HWCI 4
2	PRCV interrupt flag (proxflag)	N/A	N/A	N/A	Function Power Up

TABLE 11. *Function proximity interrupt output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Evaluation flag (evflag)	N/A	15 second intervals when PMC activated	N/A	Function Main Monitor
2	RF activation flag (didtrans)	N/A	N/A	N/A	Function Get Data
3	Timer flag (stompflag)	N/A	N/A	N/A	Function Packet Construction
4	Serial interface interrupt (H11SCCR2)	N/A	N/A	N/A	HWCI 6
5	Key line (H11PORTA, bit 4)	N/A	N/A	N/A	HWCI 6

TABLE 12. *Function time keeping input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Clock interrupt	N/A	50 Hz	N/A	HWCI 11
2	Down counter (dsec)	Seconds	N/A	Yes	Function Power Up or Main Monitor
3	General timer (dcount)	Seconds	N/A	Yes	CSCI 1 or Function Power Up
4	General timer (pcount)	1/50 Seconds	N/A	Yes	HWCI 6 or Function Power Up

TABLE 13. *Function time keeping output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Down counter (dsec)	Seconds	N/A	N/A	Function Main Monitor or Input Packet Processing
2	General timer (dcount)	Seconds	N/A	N/A	Function Transmit
3	General timer (pcount)	1/50 Seconds	N/A	N/A	Function Packet Construction
4	Interrupt flag (H11TFLG1)	N/A	N/A	N/A	HWCI 11
5	Timing register (H11TOC1)	N/A	N/A	N/A	HWCI 11
6	Timer interrupt (H11COPRST)	N/A	N/A	N/A	HWCI 11

TABLE 14. *Function A/D conversion input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	A/D conversion enable flag (adflag)	N/A	N/A	N/A	Function Power Up or Get Data
2	A/D conversion status byte (H11ADCTL)	N/A	N/A	N/A	HWCI 3
3	Wrist impedance signal (H11ADR1)	N/A	25 Hz	No	HWCI 3
4	Off wrist signal (H11ADR2)	N/A	25 Hz	No	HWCI 3
5	Battery voltage (H11ADR3)	N/A	25 Hz	No	HWCI 3

TABLE 15. *Function A/D conversion output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Wrist impedance signal (adval)	N/A	25 Hz	No	Function Get Data
2	A/D conversion completion flag (adnext)	N/A	N/A	N/A	Function Get Data
3	Off wrist signal (ad_offw)	N/A	N/A	No	Function Transmit
4	Battery voltage (ad_batt)	N/A	N/A	No	Function Transmit
5	A/D conversion control flag (H11ADCTL)	N/A	N/A	N/A	HWCI 3

TABLE 16. *Function packet construction input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Serial receiver interrupt	N/A	N/A	N/A	HWCI 6
2	Buffer input index (pin)	N/A	N/A	N/A	Function Power Up
3	Buffer output index (pout)	N/A	N/A	N/A	Function Power Up
4	General timer (pcount)	1/50 Seconds	N/A	N/A	Function Time Keeping
5	Timer flag (stompflag)	N/A	N/A	N/A	Function Proximity Interrupt
6	Individual processing flag (daemonflag)	N/A	N/A	N/A	Function Power Up or Input Packet Processing
7	Initialization subroutine (pkclear)	N/A	N/A	No	Function Power Up
8	Serial receiver status byte (H11SCSR)	N/A	N/A	N/A	HWCI 6
9	Character register (H11SCDR)	N/A	N/A	N/A	HWCI 6

TABLE 17. *Function packet construction output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Packet storage (pkbuf)	N/A	N/A	Yes	Function Input Packet Processing
2	Buffer input index (pin)	N/A	N/A	Yes	Function Input Packet Processing
3	Buffer output index (pout)	N/A	N/A	No	Function Input Packet Processing

TABLE 18. *Function input packet processing input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Packet storage buffer (pkbuf)	N/A	N/A	N/A	Function Packet Construction
2	Buffer input index (pin)	N/A	N/A	N/A	Function Packet Construction
3	Buffer output index (pout)	N/A	N/A	N/A	Function Packet Construction
4	Conversion in progress flag (adnow)	N/A	N/A	N/A	Function Power Up or Get Data
5	Down counter (dsec)	Seconds	N/A	Yes	Function Power Up or Time Keeping

TABLE 19. *Function input packet processing output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	individual processing flag (daemonflag)	N/A	N/A	N/A	Function Packet Construction
2	Record flag (reqpss)	N/A	N/A	N/A	Function Main Monitor

TABLE 20. *Function get data input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Wrist impedance signal (adval)	N/A	25 Hz	No	Function A/D Conversion
2	A/D conversion completion flag (adnext)	N/A	N/A	N/A	Function A/D Conversion
3	RF activation flag (didtrans)	N/A	N/A	N/A	Function Proximity Interrupt or Transmit

TABLE 21. *Function get data output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Digitized impedance signal (pbuf)	N/A	25 Hz	No	Function Preprocessor
2	Number of points in pbuf (npbuf)	N/A	N/A	N/A	Function Preprocessor
3	Conversion in progress flag (adnow)	N/A	N/A	N/A	Function Input Packet Processing

TABLE 22. *Function preprocessor input.*

ITFM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Digitized impedance signal (pbuf)	N/A	25 Hz	No	Function Get Data
2	Number of points in pbuf (npbuf)	N/A	N/A	N/A	Function Get Data

TABLE 23. *Function preprocessor output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Filtered data segment (pbuf)	N/A	25 Hz	No	Function Heart Rate Calculation
2	Length of data segment (seglen)	N/A	N/A	No	Function Heart Rate Calculation
3	Average value of data segment (val)	N/A	N/A	No	Function Heart Rate Calculation
4	Motion count (mocount)	N/A	15 second intervals when PMC activated	No	Function Main Monitor or Heart Rate Calculation

TABLE 24. *Function heart rate calculation input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Filtered data segment (pbuf)	N/A	25 Hz	No	Function Preprocessor
2	Length of data segment (seglen)	N/A	N/A	No	Function Preprocessor
3	Average value of data segment (val)	N/A	N/A	No	Function Preprocessor
4	Motion count (mocount)	N/A	N/A	No	Function Preprocessor
5	Stack indexes (S1, S2, S3)	N/A	N/A	N/A	Function Power Up

TABLE 25. *Function heart rate calculation output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Heart rate (brate)	Beats/Minute	15 second intervals when PMC activated	No	Function Main Monitor or Transmit

TABLE 26. *Function transmit input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Evaluation number (evnum)	N/A	N/A	No	Function Main Monitor
2	Off wrist signal (ad_offw)	N/A	N/A	No	Function A/D Conversion
3	Heart rate (hrate)	Beats/Minute	N/A	No	Function Main Monitor or Heart Rate Calculation
4	Relative motion (mocount)	N/A	N/A	No	Function Main Monitor
5	Battery voltage (ad_batt)	N/A	N/A	No	Function A/D Conversion
6	General timer (dcount)	Seconds	N/A	N/A	Function Time Keeping
7	Serial receiver status byte (H11SCSR)	N/A	N/A	N/A	HWCI 6

TABLE 27. *Function transmit output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Evaluation flag (evflag)	N/A	N/A	N/A	Function Main Monitor
2	RF activation flag (didtrans)	N/A	N/A	N/A	Function Get Data
3	Transmit packet (tpakbuf)	N/A	N/A	Yes	HWCI 6
4	Transmit packet (H11SCDR)	N/A	N/A	Yes	HWCI 6
5	Transmit key line (H11PORTA bit 4)	N/A	N/A	N/A	HWCI 6

TABLE 28. *Heart rate codes.*

HEART RATE CODE	MEANING
0	small impedance signal and no body motion
-1	processed impedance signal less than 5 seconds long
-2	heart rate greater than 240 bpm
-3	small impedance signal and body motion
-4	spectrum peak less than 350
-5	not enough beats in buffer for estimated heart rate to be valid

TABLE 29. *Acronym summary.*

ACRONYM	MEANING
CSS	Request for soldier status
DRT	Digital RF transceiver
DRTA	CSCI 1 to DRT interface
DRTB	DRT to CSCI 1 interface
DRTC	CSCI 1 to DRT interface
HHM	Hand held monitor
PMC	Personal monitor communicator
PPA	PMC positive acknowledge
PRC	Proximity receiver
PRCA	PRC to CSCI 1 interface
PSS	PMC soldier status
RTC	Real time clock
RTCA	RTC to CSCI 1 interface
VRF	Voltage Reference
VRFA	VRF to CSCI 1 interface
WZC	Wrist impedance circuit
WZCA	WZC to CSCI 1 interface
WZCB	WZC to CSCI 1 interface
WZCC	CSCI 1 to WZC interface

APPENDIX I.

10. PMC HEART RATE PROCESSING TEST

10.1 *Test Data.* Two databases of impedance signals were generated from a single PMC to test the PMC heart rate processing. The impedance signals were obtained from 11 people as they rested, stood, walked, and jogged. The impedance signals were recorded on magnetic tape and then transferred to a computer via A/D conversion. A total of approximately 10000 seconds of impedance signals were stored in the databases.

10.2 *Overview of Test.* The databases were plotted at ten second intervals. The heart rate was calculated by hand from all impedance signal plots which were not off-scale or noisy. Next, the heart rate was calculated from the digitized impedance signals using the heart rate processing software. Again the signals which were off-scale (due to too much motion) or noisy were disregarded.

10.3 *General Test Results.* A plot of software calculated heart rate (PMC calculated heart rate) versus hand calculated heart rate was obtained for each database. A correlation coefficient and a histogram of each absolute heart rate error were calculated. See Figures 8 thru 11. Based on the results, minor changes in the software were made to improve the heart rate estimation.

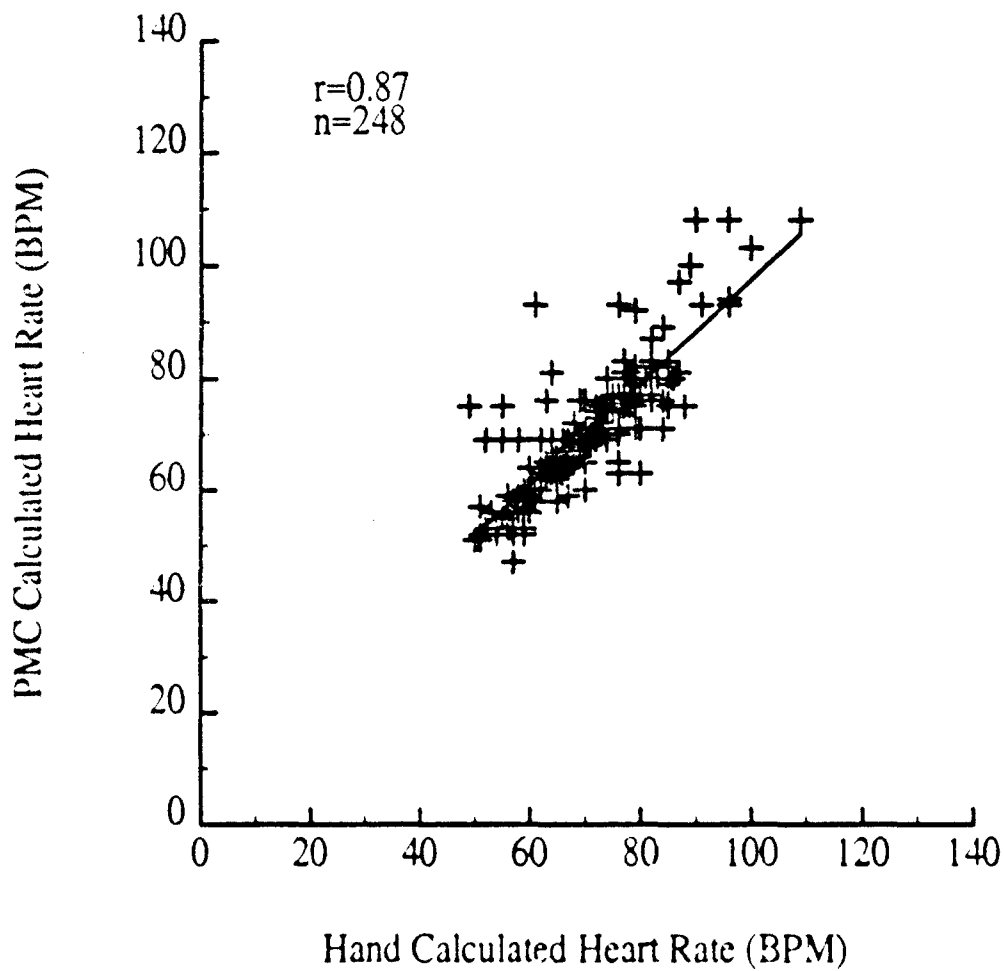


FIGURE 8. *PMC heart rate versus hand calculated heart rate, first database.*

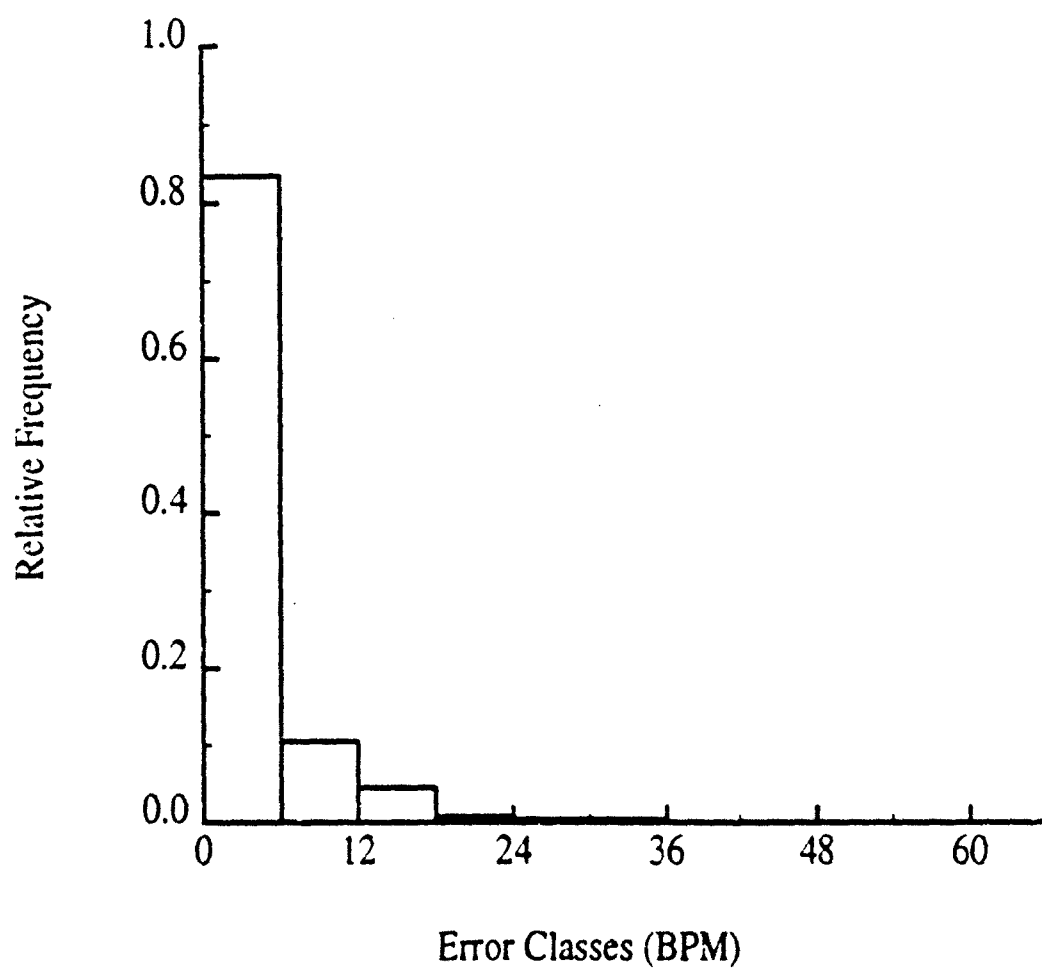


FIGURE 9. *Error histogram, first database.*

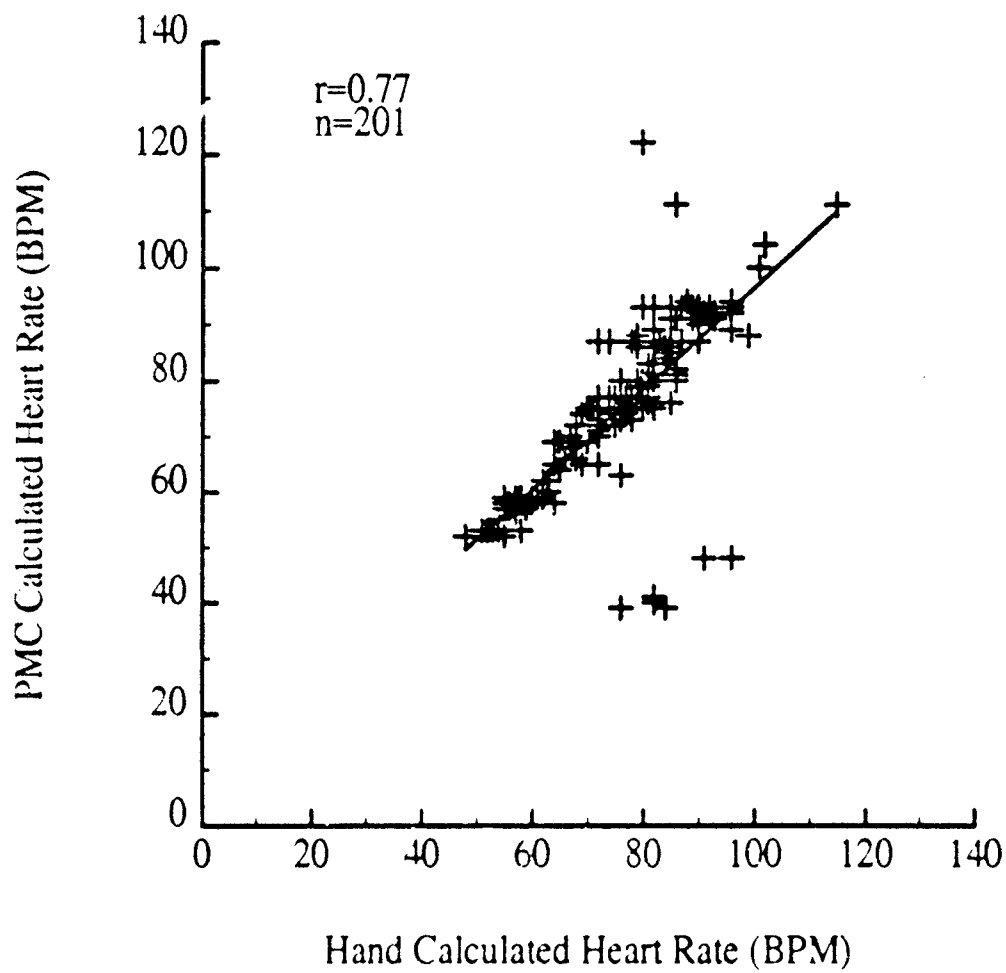


FIGURE 10. *PMC heart rate versus hand calculated heart rate, second database.*

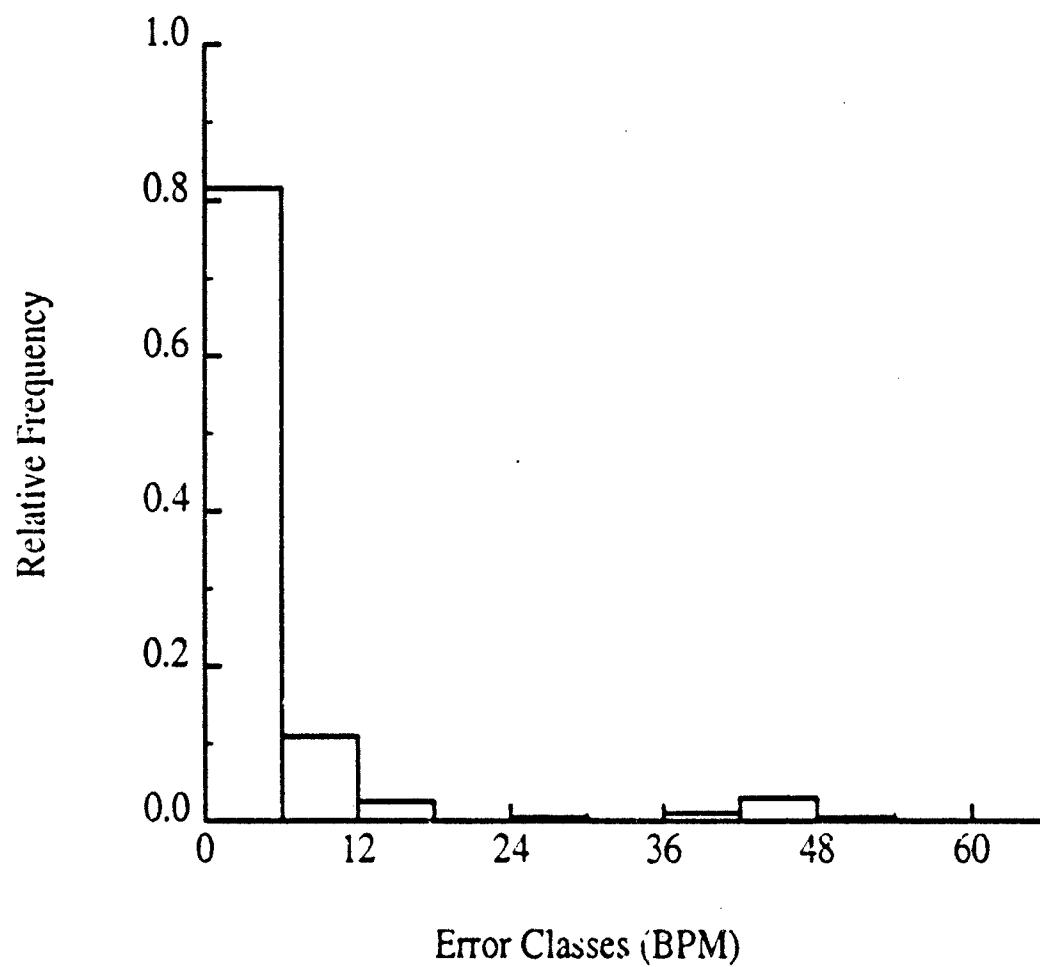


FIGURE 11. *Error histogram, second database.*

APPENDIX II.

20. PMC SOURCE CODE

```

/* PMC program */
/* written by Jim Jones, Bob Miller and Umesh Patel */
/* Purdue University */
/* 1986-1989 */

#define PMCID 13 /* must be unique for each PMC */

/* #define BENCH */

#include <stdio.h>
#include <hllreg.h>
#include <bmec.h>
#include <string.h>

#define PBUFSIZE 8 /* number of packet buffers */
#define PMASK 0x07

#define EVALTIME 60 /* Number of seconds to stay awake after evaluation */
#define SLEEPYTIME 10 /* Number of seconds to stay awake after packet */

#define MAXSAMPS 250 /* Number of a/d samples to take */

char clk50; /* 50 HZ counter */
char *pakptr; /* packet buffer pointer */
char tpakbuf[200]; /* transmit packet buffer */
char pkmess[200]; /* intermediate packet buffer */
char pkbuf[PBUFSIZE+2][50]; /* receive packet buffer */
int pin,pout; /* pkbuf input index */
int evnum,evflag; /* pkbuf output index */
int daemonflag; /* flag - insure one invocation of dopacket */
int rstate; /* current state of packet construction fsm */
int chksum,rchksum; /* packet checksums */
int dcount,dsec; /* general purpose timers */
char line[80],colstr[80]; /* character buffers */
int ad_offw; /* a/d value from off wrist circuit */
int ad_batt; /* a/d value from voltage reference */
int pcount; /* packet construction timer */
int delay; /* used in signal processing */
int mocount; /* motion counter */
int hrate; /* calculated heart rate */

```



```

int val;          /* average value */
int level;        /* stores various levels in signal processing */
int _hrate,_mcount; /* heart rate calculated in signal processing */
int batt,offw;    /* stores voltage reference and off wrist values */
int adflag;       /* flag that show that a/d conversion is on */
int adnow;        /* flag that shows that a/d values are being saved */
int pbuf[MAXSAMP]; /* buffer of wrist impedance values */
int npbuf;        /* number of points in pbuf */
int adnext;       /* flag that signals the next a/d value is ready */
int seglen;       /* data segment length */
int reqpss;       /* flag that records a request for a PSS packet */
int didtrans;     /* flag that records a radio transmission */
long int idata[256]; /* buffers for fft */
long int rdata[256];
long int spec[3][45]; /* buffers for storing spectra results */
int s1,s2,s3,countgood; /* counters for periodogram */
int adval;        /* current a/d value */
char proxflag;    /* shows proximity interrupt has occurred */
char stompflag;   /* shows radio transmitted during a/d conv */

```

```
/* function declarations */
```

```

void int_setup();
void dopacket();
__mod2__ void OC1INT();
__mod2__ void INTINT();
__mod2__ void SCIINT();

```

```

main(){
    int done();
    /* turn off transmitter */
    key(0);

    /* initialise variables */

    pin = pout = 0;
    daemonflag = 0;
    evflag = 0;
    proxflag = 0;
    stompflag = 0;
    s1 = s2 = s3 = 0;
    reqpss = 0;
    adnow = 0;
    pkclear();

    /* set up interrupts */
    int_setup();

```

```

/* set up serial port
tty_setup();
dsec = EVALTIME;

/* turn on a/d converter */
adon();
stop_enable();

/* main monitor loop */
while(1){
#ifdef BENCH
    evflag = 1;
#endif
    /* has there been a recent request for status ? */
    if(evflag){
        /* get 10 seconds of wrist impedance data */

        getdata();
        if(reqpss){
            /* send most recent status */
            dopss();
            asm("    SEI");
            reqpss = 0;
            asm("    CLI");
        }
#ifdef BENCH
        dispbuf();
#endif
        /* analyse wrist impedance data for heart rate etc. */
        getrate();
        asm("    SEI");
        hrate = _hrate;
        mocount = _mocount;
        evnum++;
        asm("    CLI");
        evflag = 0;
#ifdef BENCH
        dispbuf();
#endif
    }

    /* if there have been no recent status requests */
    /* power down radio and go to sleep */
    if(dsec == 0){
        adoff();
        rdooff();
        evflag = 0;

```

```

asm("    CLI");
H11SCCR2 = 0;
/* note: sleep instruction has been loaded at 0 */
#ifdef BENCH
asm("    JSR 0");
#endif
/* wake up */
dsec = EVALTIME;
adon();
}
}

```

/* load sleep instruction at 0 and enable sleep mode */

```

.op_enable(){
    char *p1;
    p1 = (char *) 0;
    *p1 = 0xcf;
    p1++;
    *p1 = 0x39;
    asm("    SEI");
    asm("    TPA");
    asm("    ANDA #37F");
    asm("    TAP");
    asm("    CLI");
}

```

/* turn on the a/d converter */

```

ion(){
    H11OPTION |= 0x80; /* power up a/d */
    H11PORTA |= 0x10; /* power up wrists circuit */

    adflag = 1;
}

```

/* turn off the a/d converter */

```

loff(){
    H11OPTION &= 0x7F; /* power down a/d */
    H11PORTA &= 0xEF; /* power up wrists circuit */
    adflag = 0;
}

```

/* turn off power to the radio */

```

loff(){
    H11PORTA |= 0x20;
    key(1);
}

```

turn on power to the radio */

```
on(){
  key(0);
  H11PORTA &= 0xdf;
}
```

input 10 seconds of data from the a/d converter *

```
data(){
  adnow = 1;
  do {
    asm("      SEI");
    didtrans = 0;
    asm("      CLI");
    for(npbuf=0;adnext==0;npbuf < MAXSAMP; npbuf++){
      if(didtrans == 1) npbuf = MAXSAMP;
      while(adnext==0);
      pbuf[npbuf] = adval;
      adnext = 0;
    }
    while(didtrans == 1);
  } while(adnow == 1);
  adnow = 0;
}
```

def BENCH

output pbuf to the d/a converter, bench use only */

```
obuff(){
  i,j;
  for(j=0; j<1000; j++){
    key(0);
    key(1);
    for(i=0; i < npbuf; i++){
      DAC = pbuf[i] + 128;
    }
  }
}
```

endif

transmit PPA packet *

```
pa(){
  buildpak(tpakbuf,"PPA");
  endpak(tpakbuf);
}
```

build a packet for transmission *

```
dpak(t,skhar *t,*s,{
  tcopy(t,"A");
  tsa(PMC.D,t+2);
  treat(t,"");
}
```

```

    strcat(t,s);
}

/* find the end of a string */
char *strend(s) char *s;{
    while(*s) s++;
    return(s);
}

/* encode and transmit a PSS packet */
opss(){
    evflag = 1;
    dsec = EVALTIME;
    asm("    SEI");
    offw = ad_offw;
    batt = ad_batt;
    asm("    CLI");
    strcpy(pkmess,"PSS ");
    pknum(evnum);
    pknum(offw);
    pknum(hrate);
    pknum(mocount);
    pknum(batt);
    buildpak(tpakbuf,pkmess);
    sendpak(tpakbuf);
    didtrans=1;
}

/* encode a number and add it to the
end of the packet under construction */
cnun(n) int n;{
    itoa(n,strend(pkmess));
    strcat(pkmess," ");
}

/* output a packet to the radio transmitter */
ndpak(pk) char *pk;{
    int i;
    char *p,htoc();
    p = pk;
    for(i = 0; *p; p++) i = i + *p;
    i &= 0xff;
    *p = htoc(i >> 4);
    p++;
    *p = htoc(i & 0xf);
    p++;
    *p = '0';
}

```

```

p++;
*p = '
p++;
*p = 0;
key(1);
dcount = 10;
while(dcount);
p = pk;
pkputc('
pkputc('X');
while(*p){
    pkputc(*p);
    p++;
}
pkputc(' ');
pkputc(' ');
key(0);
}

/* output a character to the radio transmitter */
pkputc(c) char c;{
    while((H11SCSR & 0x80) == 0);
    H11SCDR = c;
}

/* get a character from the radio receiver */
pkgetc(){
    while((H11SCSR & 0x20) == 0);
    return(H11SCDR);
}

/* clear the input packet buffer */
pkclear(){
    pkptr = pkbuf;
    chksum = 0;
    rotate = 0;
}

/* service the proximity receiver interrupt */
mod2 void INTINT(){
    if('proxflag){
        proxflag = 1;
        stopflag = 1;
        asm("CLI");
        H11SCCR2 = 0x20;
        edon();
    }
}

```

```

/* send PPA packet */
doppa();
evflag = 1;
didtrans = 1;
asm("    SEI");
proxflag = 0;
asm("    CLI");
}

}

* key the radio transmitter */
key(i) int i;{
    if(i == 0)H11PORTA |= 0x08;
    else H11PORTA &= 0xF7;
}

* service the serial interface receive interrupt */
* SCIINT is entered when a character is received */
mod2__ void SCIINT(){
    int c;
    char *s;
    c = pkgetc();
    /* finite state machine for packet construction */
    switch(rstate){
        case 0:
            /* look for packet start character " */
            if(c == '"'){
                *pakptr++ = c;
                chksum = c;
                rstate = 1;
            }
            break;
        case 1:
            /* look for second start character A */
            if(c == '"')break;
            if(c == 'A'){
                *pakptr++ = c;
                chksum += c;
                rstate = 2;
                pcount = 50;
            }
            else pkclear();
            break;
        case 2:
            if(c == '"'){
                pkclear();
                *pakptr++ = c;
            }
    }
}

```

```

    chksum = c;
    rstate = 1;
}
/* look for packet end characters */
else if((c == '0') || (c == ' ')){
    *pakptr = 0;
    s = pakptr;
    s--; s--;
    rchksum = xtoi(s);
    chksum -= *s++;
    chksum -= *s;
    chksum &= 0xff;
    if(chksum == rchksum){
        pin = (pin + 1) & PMASK;
        pkclear();
        if(daemonflag == 0){
            dopacket();
        }
    }
    else{
        pkclear();
        asm("    CLI");
    }
}
/* add a character to the packet */
else {
    if((pakptr >= &pkbuf[pin][49]) || (pcount == 0))pkclear();
    else{
        *pakptr++ = c;
        chksum += c;
    }
}
break;
}
}

/* process input packet */
void dopacket(){
    char *s,num[10];
    int ndigit,id;
    asm("    SEI");
    daemonflag = 1;
    while(pin != pout){
        asm("    CLI");
        /* process pkbuf[pout] */
        s = pkbuf[pout];
        s++; s++;

```



```

for(ndigit=0; isdigit(s[ndigit]) && ndigit < 9; ndigit++) num[ndigit] = s[ndigit];
num[ndigit] = ' ';
id = atoi(num);
if(id == PMCID || id == 0){
    s = s + ndigit + 1;
    do {
        stompflag = 0;
        if(dsec < SLEEPYTIME) dsec = SLEEPYTIME;
        /* recognize request for status */
        if(strncmp(s, "CSS", 3) == 0){
            if(!adnow || (s[4] == 'I')) dopss();
            else reqpss = 1;
        }
    } while(stompflag);
}
asm("    SEI");
pout = (pout + 1) & PMASK;
}
daemonflag = 0;
asm("    CLI");
}

/* service OC1 (real time clock) interrupt */
__mod2__ void OC1INT(){
    H11TFLG1 = 0x80;
    H11TOC1 += 0x9c40;
    H11COPRST = 0x55;
    H11COPRST = 0xaa;

    clk50++;

    /* activate a/d conversion on odd clock ticks */

    if(adflag && (clk50 & 1)){
        H11ADCTL = 0x10;
        while(H11ADCTL & 0x80 == 0);
        adval = (int)H11ADR1 - 128;
#ifdef BENCH
        /* DAC = H11ADR1 ; */
#endif
        adnext = 1;
        ad_offw = H11ADR2;
        ad_batt = H11ADR3;
    }
    if(clk50 >= 50){
        clk50 = 0;
    }
}

```

```

        if(dsec)dsec--;
    }
    if(dcount)dcount--;
    if(pcount)pcount--;
}

/* service COP interrupt */
__mod2__ void COPINT(){
    asm("    JMP $C000");
}

/* set up interrupt service routines */
void int_setup(){
    __mod2__ void OC1INT();
    __mod2__ void INTINT();
    __mod2__ void COPINT();
    __mod2__ void SCIINT();
    void VECTOR();
    dcount = 0;
    pcount = 0;
    VECTOR(OC1INT,9);
    VECTOR(INTINT,14);
    VECTOR(COPINT,17);
    VECTOR(COPINT,18);
    VECTOR(SCIINT,0);
    H11TOC1 = 0;
    H11TMSK1 |= 0x80;
}

/* set up serial interface */
void sty_setup(){
    chksum = 0;
    rechsum = 0;
    rstate = 0;
    H11SCCR1 = 0;
    H11SCCR2 = 0x2C;
    H11BAUD = 0x35;
    H11DDRD = 2;
    H11PORTD = 0;

#include "hex.c"
#include "filter.c"
#include "sigproc.c"

```

* hex.c program

*/

* convert hex ascii string to integer */

```
atoi(s)char *s;{
    int i;
    i=ctoh(s[0]);
    i=(i<<4) | ctoh(s[1]);
    return(i);
```

* convert hex ascii digit to integer */

```
ctoh(c1)
int c1;
    if(c1>96)c1=c1-32;
    c1=c1-48;
    if(c1>9)c1=c1-7;
    return(c1);
```

* convert integer to hex ascii character */

```
char htoc(i) int i;{
    char h;
    h = i + '0';
    if(h > '9')h += 7;
    return(h);
}
```

```

/* filter.c program */

/* find the median value in a buffer */
median(buf, n) int *buf; int n; {
    int i, j, m, tmp;

    m = n / 2;
    for(j = n; j > m; j--){
        for(i = 0; i < j - 1; i++){
            if(buf[i] > buf[i + 1]){
                tmp = buf[i];
                buf[i] = buf[i + 1];
                buf[i + 1] = tmp;
            }
        }
    }
    return(buf[m]);
}

/* 0.5 HZ high pass filter */
#define HPWIDTH 10
hpf() {
    int i, j, sum;

    for(i=0; i < npbuf - HPWIDTH; i++){
        sum = 0;
        for(j=0; j < HPWIDTH; j++) sum += pbuf[i+j];
        pbuf[i] -= sum / HPWIDTH;
    }
    delay = delay + HPWIDTH;
    npbuf -= HPWIDTH;
}

/* 4.0 HZ low pass filter (FIR) */
lpf(){
    static int c[25] = {0.0,0.0,2.2,-1,-10,-12,4,37,76,94,
                        76,37,4,-12,-10,-1,2,2,0,0,0,0};
    int i, j, sum;
    for(i=13; i < (npbuf - 12); i++){
        sum = 0;
        for(j = 0; j < 25; j++) sum = sum + (pbuf[i-13+j] * c[j]);
        pbuf[i-13] = sum >> 8;
    }
    npbuf -= 25;
}

```

```

/* sigproc.c program */

#include "sigproc.h"

getrate(){
    delay = 0;
    _mocount = doseg(100,0,npbuf,mseg,&nmseg);

/* CONSIDER DATA SEGMENTS UNDER 5 SEC. LONG AS MOTION */
/* TRIM AND ZERO-OUT MOTION SEGMENTS */

    fill(0);
/* FILTER DATA - FIND AVERAGE VALUE */
    lpf();
    hpf();
    val = avgval();
#ifdef BENCH
    sprintf(line,"avgval %d",val);
/* sendpak(line); */
#endif
/* REPEAT WITH NEW THRESHOLD LEVEL */
    if(val < 1){
        if(_mocount == 0)_hrate = 0;
        else _hrate = -1;
    }
    else {
        level = getlevel() * 5;
        if(level < 25)level = 25;
        if(level > 75)level = 75;
        shift(mseg,nmseg);
        fill(80);
        _mocount = _mocount + doseg(level,0,npbuf,mseg,&nmseg);
        seglen = maxmseg();
        if(seglen > 125){
            countgood = stack(1);
            trim();
            fill(0);
#ifdef BENCH
            dispbuf();
#endif
            acorr();
            if(_hrate > 240)_hrate = -2;
        }
        else{
            _hrate = -1;
        }
    }
}

```

```

    if(_hrate <= 0)countgood = stack(0); /* reset countgood to 0 */
    if(_hrate <= 0)countgood = stack(0);
    if(_hrate <= 0)countgood = stack(0);
}

getlevel(){ /* find median value of signal */
    int i,*b,j,n;
    b = (int *)rdata;
    for(i=0; i < nmseg; i++){
        if(mseg[i].on){
            n = mseg[i].fn - mseg[i].strt + 1;
            for(j=0; j<n; j++)b[j] = abs(pbuf[j+mseg[i].strt]);
            return(median(b,n));
        }
    }
}

shift(ms,nm)struct segment ms[]; int nm;{
    int i;
    for(i=0; i<nm; i++){
        ms[i].strt = (ms[i].strt - delay) > 0 ? ms[i].strt - delay : 0;
        ms[i].fn = (ms[i].fn - delay) > 0 ? ms[i].fn - delay : 0;
    }
}

maxmseg(){ /* calculates maximum segment length */
    int i;
    for(i=0; i < nmseg; i++){
        if(mseg[i].on)return(mseg[i].fn - mseg[i].strt + 1);
    }
    return(0);
}

/* fill in val when mseg[i].on == 0 (off-scale) */
/* trapezoidal window on-scale segment */
fill(val) int val; {
    int i,j;
    for(i=0;i<nmseg;i++){
        if (mseg[i].on == 0) for (j=mseg[i].strt;j<=mseg[i].fn;j++) pbuf[j] = val;
        else pwindow(mseg[i].strt,mseg[i].fn-mseg[i].strt+1);
    }
}

trim(){ /* trim ends of data segments */
    int i,j;
    for(i=0;i<nmseg;i++){
        if (mseg[i].on){
            j = mseg[i].strt;
            if(pbuf[j] > 0)for(;j<=mseg[i].fn && pbuf[j] > 0;j++) pbuf[j] = 0;
        }
    }
}

```

```

    if(pbuf[j] < 0)for(;j<=mseg[i].fin && pbuf[j] < 0;j++) pbuf[j] = 0;
    mseg[i].strt = j;
    j = mseg[i].fin;
    if(pbuf[j] > 0)for(;j>mseg[i].strt && pbuf[j] > 0;j--) pbuf[j] = 0;
    if(pbuf[j] < 0)for(;j>mseg[i].strt && pbuf[j] < 0;j--) pbuf[j] = 0;
    mseg[i].fin = j;
}
}
}

avgval(){ /* returns average value of data segment */
    int i,sum;
    for(i=0,sum=0;i<npbuf;i++)
        sum = sum + abs(pbuf[i]);
    return(sum / npbuf);
}

```

/* based on value v, determines on and off scale segments */

```

doseg(v,a,b,n) int v,a,b; struct segment s[]; int *n;{

```

```

    int i,on,nseg,new;
    int ntrans;
    ntrans = 0;
    nseg = 0;
    *n = 0;
    s[*n].strt = a;
    on = (pbuf[a] < v)&&(pbuf[a] > -v);
    s[*n].on = on;
    *n = 0;
    nseg = 1;
    for(i=a+1; i < b; i++){
        nseg++;
        new = (pbuf[i] < v)&&(pbuf[i] > -v);
        if(on && !new){
            ntrans++;
            if(nseg > 125){
                s[*n].fin = i-1;
                (*n)++;
                s[*n].strt = i;
                on = new;
                s[*n].on = on;
                nseg = 1;
            }
        }
        else {
            on = new;
            s[*n].on = on;
            if(*n > 0){
                if(s[*n-1].on == on){
                    (*n)--;
                }
            }
        }
    }
}

```

```

        nseg = nseg + s[*n].fin - s[*n].strt + 1;
    }
}

if(new && !on){
    ntrans++;
    s[*n].fin = i-1;
    (*n)++;
    s[*n].strt = i;
    on = new;
    s[*n].on = on;
    nseg = 1;
}

s[*n].fin = b-1;
if(s[*n].on && nseg < 125 && *n > 0)
    if(s[*n-1].on == 0){
        (*n)--;
        s[*n].fin = b-1;
    }
(*n)++;
return(ntrans);
}

window(s,n){ /* trapezoidal window used on data segment */
    int i;
    for(i=0; i<25; i++){
        pbuf[i+s] = (pbuf[i+s]*i)/25;
        pbuf[n-i-1+s] = (pbuf[n-i-1+s]*i)/25;
    }
}

t(pr,pi,m,n,sign) /* integer FFT */
long int *pr,*pi;
int m,n,sign;{

    static int icos[11] = {0,-1024,0,723,946,1005,1019,1023,1024,1024,1024};
    static int isin[11] = {0,0,1024,723,392,200,100,50,26,12,6};
    long int ur_n,d,ul_n,wr_n,wl_n,tr_n,tl_n;
    long int pitmp,prtmp;
    int i,j,k,l,nv2,nm1,ip,le1,le;
    nm1 = n-1;
    nv2 = n >> 1;
    j = 0;
    d = 1024;

```



```

for (i=0; i<nm1; i++) {
    if (i < j) {
        swap(pr-i,pr-j);
        swap(pi-i,pi-j);
    }
    for(k=nv2;k<=j;j=j-k,k=k>>1);
    j -= k;
}

```

```

for (l=1; l<=m; l++) {
    le = 1 << l;
    le1 = le>>1;
    ur_n = 1024;
    ul_n = 0;
    wr_n = icos[l];
    wl_n = isin[l] * sign;
    for (j = 0; j < le1; j++) {
        for (i = j; i < n; i += le) {
            ip = i + le1;
            tr_n = pr[ip] * ur_n - pi[ip] * ul_n;
            ti_n = pi[ip] * ur_n + pr[ip] * ul_n;
            prtmp = pr[ip] << 10;
            pitmp = pi[ip] << 10;
            pr[ip] = (prtmp-tr_n)>>10;
            pi[ip] = (pitmp-ti_n)>>10;
            pr[ip] = (prtmp-tr_n)>>10;
            pi[ip] = (pitmp-ti_n)>>10;
        }
        tr_n = ur_n;
        ur_n = (ur_n*wr_n - ul_n*wl_n)>>10;
        ul_n = (tr_n*wl_n + ul_n*wr_n)>>10;
    }
}
if (sign <= 0) return;
for (i=0; i<n; i++) {
    pr[i] = pr[i]^m;
    pi[i] = pi[i]^m;
}

```

```

swap(px,py) long int *px,*py;
long int temp;

```

```

temp = *px;
*px = *py;
*py = temp;

```

```

rrr) { /* calculate spectrum of signal */
  at a,p,ij0;
  long int *ppr,*ppi,max0,tr,ti,a,b;

  def BENCH
  long mfact;
  ndif

  ori=0;i<npbuf;i++) rdata[i] = pbuf[i];
  ori=npbuf;i<256;i++) rdata[i] = 0;
  ori=0;i<256;i++) ldata[i] = 0;

  pr = rdata;
  pi = ldata;
  = 9;
  = 256;
  t(ppr,ppi,p,n,-1);
  pbuf = 128;
  = 128;
  ori=0;i<n;i++){ /* calculate spectrum */
    tr = rdata[i];
    ti = ldata[i];
    rdata[i] = (tr*tr + ti*ti) >> 8;
    ldata[i] = 0;
  }
  adspec(); /* Update and load new data into */
            /* storage arrays for periodogram averaging */

  to periodogram averaging */
  if (countgood == 1){
    }
  else if (countgood == 2){
    for(i=0;i<45;i++){
      rdata[i] = (rdata[i]+spec 1[i]) >> 1;
    }
  }
  else if (countgood == 3){
    for(i=0;i<45;i++){
      rdata[i] = (spec 1[i]+spec 2[i]+rdata[i])*3;
    }
  }
  nd of periodogram averaging */

  nd frequency where peak of spectrum occurs */
  mit search to 30 to 240 bpm */
  ax0 = 0;
  = 0;

```

```

for(i=5; i<42; i++){
    if(rdata[i-1] < rdata[i] && rdata[i] >= rdata[i+1] && rdata[i] > max0){
        max0 = rdata[i];
        j0 = i;
    }
}

/* calculate heart rate based on that frequency */
a = rdata[j0] + rdata[j0-1] + rdata[j0+1];
b = (long)rdata[j0]*j0 + (long)rdata[j0-1] * (j0-1) + (long)rdata[j0+1] * (j0+1);
b = (b * 10) / a;
_hrate = (b * 75) / 128;
if(max0 < 350) _hrate = -4; /* -4 means spectrum is too small */
/* a check to omit low heart rate answers for short data segments */
if((_hrate < 49 && seglen < 167) || (_hrate < 43 && seglen < 192)
    || (_hrate < 37 && seglen < 210)) _hrate = -5;

#ifdef BENCH
    i = max0;
    sprintf(line, "max0 %d", i);
    /*sendpak(line);*/
#endif

#ifdef BENCH
    mfact = max0 / 128;
    for(i=0; i<n; i++){
        pbuf[i] = rdata[i] / mfact * 100;
        if(pbuf[i] < 0) pbuf[i] = 0;
    }
#endif

updatepec(){ /* updating saved spectrums */
    int i;
    for (i=0; i<45; i++){
        spec 2 [i] = spec 1 [i];
        spec 1 [i] = spec 0 [i];
        spec 0 [i] = rdata[i];
    }
}

tach(i) int i; /* counter routine to update */
int sum; /* number of good data segments */
    s1 = s2;
    s2 = s3;
    s3 = i;
    sum = s1 + s2 + s3;
    return sum;

```

PART II OF TWO PARTS

SOFTWARE REQUIREMENTS SPECIFICATION
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A005

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

CONTENTS

Paragraph	Page
1. SCOPE.....	7
1.1 Identification.....	7
1.2 Purpose.....	7
1.3 Introduction.....	7
2. APPLICABLE DOCUMENTS.....	7
2.1 Government Documents.....	7
2.2 Non-Government Documents.....	7
3. REQUIREMENTS.....	8
3.1 Programming Requirements.....	8
3.1.1 Programming Language(s).....	8
3.1.2 Compiler/Assembler.....	8
3.1.3 Programming Standards.....	8
3.2 Design Requirements.....	8
3.2.1 Sizing and Timing Requirements.....	8
3.2.2 Design Standards.....	8
3.2.3 Design Constraints.....	9
3.3 Interface Requirements.....	9
3.3.1 Interface Relationships.....	9
3.3.2 Interface Identification and Documentation.....	9
3.3.3 Detailed Interface Requirements.....	9
3.3.3.1 CSCI-to-CSCI Interface Requirements.....	9
3.3.3.2 CSCI-to-HWCI or Critical Item Requirements.....	9
3.3.3.2.1 Voltage Reference Interface.....	9
3.3.3.2.2 Digital RF Transceiver Interfaces.....	9
3.3.3.2.3 Display Interface.....	10
3.3.3.2.4 Activate Switch Interface.....	11
3.3.3.2.5 Proximity Transmitter Interface.....	11
3.3.3.2.6 Real Time Clock Interface.....	11

CONTENTS - continued

Paragraph	Page
3.4 Detailed Functional and Performance Requirements	12
3.4.1 Power Up Initialization Function	12
3.4.1.1 Inputs.....	12
3.4.1.2 Processing	12
3.4.1.3 Outputs.....	13
3.4.2 Main Monitor Function	13
3.4.2.1 Inputs.....	13
3.4.2.2 Processing	13
3.4.2.3 Outputs.....	14
3.4.3 Clock Interrupt Function	14
3.4.3.1 Time Keeping Function.....	14
3.4.3.1.1 Inputs	14
3.4.3.1.2 Processing	14
3.4.3.1.3 Outputs.....	15
3.4.3.2 A/D Conversion Function	15
3.4.3.2.1 Inputs	15
3.4.3.2.2 Processing	15
3.4.3.2.3 Outputs.....	16
3.4.4 Serial Receive Function	16
3.4.4.1 Packet Construct Function	16
3.4.4.1.1 Inputs	16
3.4.4.1.2 Processing	16
3.4.4.1.3 Outputs.....	16
3.4.4.2 Input Packet Processing Function.....	17
3.4.4.2.1 Inputs	17
3.4.4.2.2 Processing	17
3.4.4.2.3 Outputs.....	17
3.4.5 Display Function	17
3.4.5.1 Inputs.....	17
3.4.5.2 Processing	18
3.4.5.3 Outputs.....	18
3.4.6 Transmit Function	18
3.4.6.1 Inputs.....	18
3.4.6.2 Processing	18
3.4.6.3 Outputs.....	19

CONTENTS - continued

Paragraph	Page
3.5	Adaptation Requirements.....19
3.5.1	System Environment.....19
3.5.2	System Parameters.....19
3.5.3	System Capacities.....19
3.6	Quality Factors.....19
3.6.1	Correctness Requirements.....19
3.6.2	Reliability Requirements.....19
3.6.3	Efficiency Requirements.....19
3.6.4	Integrity Requirements.....19
3.6.5	Usability Requirements.....19
3.6.6	Maintainability Requirements.....19
3.6.7	Testability Requirements.....20
3.6.8	Flexibility Requirements.....20
3.6.9	Portability Requirements.....20
3.6.10	Reusability Requirements.....20
3.6.11	Interoperability Requirements.....20
3.6.12	Additional Quality Factor Requirements.....20
3.7	CSCI Support.....20
3.7.1	Facility Requirements.....20
3.7.2	Equipment Requirements.....20
3.7.3	Software Requirements.....21
3.7.4	Facility Requirements.....21
3.8	Traceability.....21
4.	QUALIFICATION REQUIREMENTS.....21
4.1	General Qualification Requirements.....21
4.2	Special Qualification Requirements.....21
5.	PREPARATION FOR DELIVERY.....21
5.1	Preparation for Delivery.....21
6.	NOTES.....21
6.1	Acronyms.....21

CONTENTS - continued

FIGURES

Figure		Page
1.	Interface block diagram.....	22
2.	Function block diagram	23
3.	Function main monitor flowchart.....	24
4.	Function packet construction finite state machine	25

TABLES

Table		Page
1.	Interface identification/documentation	26
2.	Interface summary	27
3.	Request for soldier status (CSS) format	28
4.	PMC positive acknowledge (PPA) format.....	29
5.	PMC soldier status (PSS) format	30
6.	Function power up initialization input.....	31
7.	Function power up initialization output.....	32
8.	Function main monitor input.....	34
9.	Function main monitor output.....	35
10.	Function time keeping input	36
11.	Function time keeping output	37
12.	Function A/D conversion input.....	38
13.	Function A/D conversion output.....	39
14.	Function packet construction input.....	40
15.	Function packet construction output.....	41
16.	Function input packet processing input.....	42
17.	Function input packet processing output	43
18.	Function display input	44
19.	Function display output	45
20.	Function transmit input.....	46
21.	Function transmit output.....	47
22.	Acronym summary	48

CONTENTS - continued

APPENDIX

Appendix

Page

Appendix I

10.	HHM Source Code	49
-----	-----------------------	----

1. SCOPE

1.1 *Identification.* This Software Requirements Specification establishes the requirements for the CSCI identified as Hand Held Monitor Software, HHM-CSCI.2, of the Personal Monitor and Communicator, PMC-SYS.1, System.

1.2 *Purpose.* The Personal Monitor and Communicator System provides a means of remotely monitoring the heart rate and body motion of a soldier. The HHM-CSCI.2 drives the hand held monitor. This unit consists of a 2-way radio link, a microprocessor, and a LCD display. After receiving a digital signal, the software processes the information and displays the computed heart rate of the monitored soldier.

1.3 *Introduction.* This document provides detailed and complete specifications of the software developed to drive the hand held monitor unit of the Personal Monitor and Communicator System.

2. APPLICABLE DOCUMENTS

2.1 *Government Documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

SPECIFICATIONS

Military

MIL-S-83490 Specifications, Types and Forms

STANDARDS

Military

DOD-STD-2167 Defense System Software Development

MIL-STD-483A Configuration Management Practices For
Systems, Equipment, Munitions, and Computer
Programs

MIL-STD-490A Specifications Practices

2.2 *Non-Government Documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

OTHER PUBLICATIONS

Oppenheim, A.V., Schafer, R.W.
Digital Signal Processing
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975

Kernighan, B.W., Ritchie, D.M.
The C Programming Language
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1978

Introl C MS-DOS Host Guide Version 1.0
Introl Corporation, Milwaukee, WI, 1987

Motorola Semiconductor Technical Data
Advance Information for the MC68HC11A8
HCMOS Single-Chip Microcomputer
Order number MC68HC11A8/D

Programmer's Reference Manual
M68HC11 HCMOS Single-Chip Microcomputer
Order number M68HC11PM/AD

3. REQUIREMENTS

3.1 *Programming Requirements.*

3.1.1 *Programming Languages.* Two programming languages are implemented for this CSCI. The languages are C and Motorola 68HC11 assembler.

3.1.2 *Compiler/Assembler.* The C compiler is written by Introl Corporation and is version C-11 of the Introl-C Cross-Compiler Systems series.

3.1.3 *Programming Standards.* Structured programming techniques are followed.

3.2 *Design Requirements.*

3.2.1 *Sizing and Timing Requirements.* The CSCI uses all of the memory available in the RAM. In addition, the CSCI uses 6810 bytes of PROM. The CSCI processes incoming data at a rate of 300 baud (real time serial communication).

3.2.2 *Design Standards.* Development of this CSCI follows the design standards of the

William A. Hillenbrand Biomedical Engineering Center.

3.2.3 *Design Constraints.* The CSCI is limited by available memory and the implementation of integer math.

3.3 *Interface Requirements.*

3.3.1 *Interface Relationships.* See interface block diagram (Figure 1).

3.3.2 *Interface Identification and Documentation.* See interface identification table (Table 1.)

3.3.3 *Detailed Interface Requirements.*

3.3.3.1 *CSCI-to-CSCI Interface Requirements.* This section is not applicable to this specification.

3.3.3.2 *CSCI-to-HWCI or Critical Item Requirements.*

3.3.3.2.1 *Voltage Reference Interface.*

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The signal will have a numerical value in the range of 0 to 255. Signals with values greater than or equal to 68 indicate the HHM batteries are low (< 4.5 volts).
- c. *Memory Buffer And Location.* This section is not applicable to this interface.
- d. *Transfer Protocol.* A value of 0x10 (0x indicates the number to follow is hexadecimal) is sent to the A/D control register, H11ADCTL. When bit 7 of the register goes high (1) register H11.ADR3 is read to determine the voltage level of the HHM batteries.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.2 *Digital RF Transceiver Interfaces.*

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The transmit data interface (IF DRTD) sends one message. The format of this message, a request for soldier status (CSS), is shown in Table 3. The receive data interface (IF DRTE) is capable of receiving two messages. The formats of these messages, PMC positive acknowledge (PPA)

and PMC soldier status (PSS), are given in Tables 4 and 5.

- c. *Memory Buffer And Location.* The memory buffer necessary for the digital RF transceiver consists of one 200-byte output buffer, "tpakbuf," and one 50-byte input buffer, "pkbuf."
- d. *Transfer Protocol.* Serial communication is used in both the transmit and receive data protocols. To transmit a message, the serial communication status register, H11SCSR, is polled until the transmitter-ready bit goes high (1). The character to be transmitted is then written to the serial communication data register, H11SCDR. To receive a message, the serial communication receiver interrupts the processor when a character is received. The interrupt routine implements a finite state machine to recognize valid packets.
- e. *Initiation Condition.* Table 2 indicates general conditions for the initiation of all three signals pertaining to the digital RF transceiver interface. The DRTD interface can send one message, the CSS message. The digital RF transceiver will broadcast this message to the activated PMC. The DRTE interface receives two messages, the PPA message and the PSS message. The PPA message is received after the HHM has activated a selected PMC. This message verifies the selection and activation. The PSS message is the PMC's response to a prompt from the HHM for information.
- f. *Priority Level.* The receive data and the transmit data sequentially alternate having the highest priority.
- g. *Expected Response.* Table 2 shows the expected responses for all three signals. The receive data interface (DRTE IF) is the only interface with timing restrictions. The DRTE interface receives and processes characters only when a message is expected. The maximum response time per character is approximately 1/30 second which is the time necessary for character transmission at 300 baud. If incoming characters are not responded to in time, then the receive character buffer will be overrun causing the character to be lost. The message will need to be retransmitted. This is not fatal unless characters are lost in every packet.

3.3.3.2.3 Display Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The display signal is based on digital logic levels.
- c. *Memory Buffer And Location.* The memory buffer, variable "line," consists of 80 bytes. The message strings displayed are stored in PROM. In addition, the registers DISPLAY and DISPDAT occupy two consecutive memory locations.

- d. *Transfer Protocol.* Data may be written to either display register, DISPLAY or DISPDAT. To write a byte of data, bit 7 of the register DISPLAY is polled until it goes high (1). Then the data byte may be written to either register.
- e. *Initiation Condition.* The interface is initiated when the HHM receives an activate signal, either PMC message (PPA or PSS), or a low battery voltage message. See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.4 Activate Switch Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* When the activate switch is pressed, a digital input signal is transmitted to bit 2 of Port A of the 68HC11.
- c. *Memory Buffer And Location.* This section is not applicable to this interface.
- d. *Transfer Protocol.* The digital input signal from the activate switch is polled 50 times a second by an interrupt routine driven by the real time clock.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.5 Proximity Transmitter Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The signal sent by CSCI.2 to the proximity transmitter is the digitized output from bit 5 of Port A of the 68HC11.
- c. *Memory Buffer And Location.* This section is not applicable to this interface.
- d. *Transfer Protocol.* To activate the proximity transmitter, bit 5 of port A is set to high (1) by CSCI.2.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* See interface table (Table 2).

3.3.3.2.6 Real Time Clock Interface.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The signal is an interrupt that is emitted every 20 milliseconds.
- c. *Memory Buffer And Location.* This section is not applicable to this interface.

- d. *Transfer Protocol.* This section is not applicable to this interface.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* The system's signal causes internal interrupts at 50 Hz. The generated interrupts are necessary for the A/D conversions, the battery voltage checks, and the functions which use elapsed time counters and count-down timers. See interface table (Table 2).

3.4 Detailed Functional and Performance Requirements.

3.4.1 Power Up Initialization Function.

3.4.1.1 *Inputs.* The function has 1 input which is the power up reset. The reset serves as a hardware and software initialization. See the table for the power up function input (Table 6) for additional information.

3.4.1.2 Processing.

- a. *Intent.* This function initializes the 68HC11 microcomputer hardware, the interrupt vectors, and the program variables.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The six independent initializations in this function are described as follows. All program variables are assigned values of zero. The routine "key" turns off the key line to the transmitter as soon as possible since the initial hardware power-up causes the transmitter to be keyed. Routine "int_setup" initializes the interrupt vectors for the interrupt routines OCINT, INTINT, and COPINT. The routine also initializes output compare 1, and enables the timer interrupt. Routine "tty_setup" has a three fold purpose. The routine sets the serial interface for one start bit and eight data bits (H11SCCR1 = 0), enables the transmitter, receiver, and receiver interrupt (H11SCCR2 = 2C), and sets the baud register for 300 baud (H11BAUD = 0x35). The routine "adon" turns on the A/D converter. The LCD display is initialized and cleared by the routine "init_disp."
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.1.3 Outputs. The function has 14 initialized outputs. The flag "adflag" is set to 1 to indicate the A/D converter is enabled and the flag "actflag" is set to 0 to show that no PMC has yet been activated. The initializing subroutine "pkclear" clears the buffer area necessary for packet construction. The variables "pin" and "pout" are input and output index pointers set to the location of the stored packet information. The user is informed the unit is attempting to activate a PMC by the output message sent to the LCD display. The other 8 outputs are hardware oriented. The baud rate register, H11BAUD, is set for a rate of 300 baud. The port D direction register, H11DDRD, is initialized to 2. The A/D converter, via the H11OPTION register, is powered up. The key line, H11PORTA (bit 4), is set inactive. Both the serial interface register, H11SCCR1, and the output compare 1 register, H11TOC1, are cleared. And the serial interface interrupt, H11SCCR2, and the output compare interrupt, H11TMSK1, are enabled. See the table for the power up function output (Table 7) for additional information.

3.4.2 Main Monitor Function.

3.4.2.1 Inputs. The function has 7 inputs. The variable "adval" contains the battery voltage reference. The variable "id" stores the identification code of the PMC with which the HHM is communicating. The register status indicates whether the expected information packet has been received. The two counters are "dcount", which is a general counter and "etime", which indicates the time elapsed between HHM-PMC communications. The flag "actflag" shows whether the PMC has been activated. The structure (record) 'opp' contains all of the decoded packet information. See the table for the main monitor function input (Table 8) for additional information.

3.4.2.2 Processing.

- a. *Intent.* This function monitors the activate button of the HHM which powers up a PMC. When the button is pressed, the main monitor executes the activation sequence for the PMC. Once the PMC is activated, the main monitor requests and displays the PMC status every 15 seconds. This process is terminated when another activation request is received, unrecoverable communication errors occur, or the HHM is turned off.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The function is a continuous loop which checks the HHM activate button. Pressing the button activates the PMC and the evaluation sequence is begun. The sequence, which is repeated every 15 seconds, consists of requesting PMC status information and displaying the results on the HHM. See the main monitor flow chart (Figure 3) for additional detail.

- d. *Error Detection and Recovery.* If errors are detected while trying to activate the PMC or attempting to receive the PMC status then the following procedure is performed. The attempt is retried up to 10 times. The time delay between retries varies as the delay depends on the retry number. If after 10 tries the communication is unsuccessful, then all attempts cease. It is up to the user to try again.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.2.3 *Outputs.* The function has 6 outputs. The variable "retries" stores the number of attempts which have been made to establish a communication channel between the HHM and the selected PMC. The variable "id" contains the PMC's identification code. The two counters input to the function, "etime" and "dcount" are updated and passed on to the time keeping function. The flag, "actflag," shows the activation status. Messages are transmitted to the LCD display from this function to keep the user informed of the communication status. See the table for the main monitor function output (Table 9) for additional information.

3.4.3 *Clock Interrupt Function.*

3.4.3.1 *Time Keeping Function.*

3.4.3.1.1 *Inputs.* The function has 4 inputs. The clock interrupt provides for accurate time keeping. The general counter "dcount" is used to time delays while the counter "etime" is used to measure elapsed communication time. And the flag "actflag" indicates if a PMC has been activated. See the table for the time keeping function input (Table 10) for additional information.

3.4.3.1.2 *Processing*

- a. *Intent.* The function provides a regular interrupt at a frequency of 50 Hz for various time keeping functions.
- b. *Parameters Affected.* The output compare 1 register of the 68HC11 is continuously updated to permit an interrupt to occur every 20 milliseconds.
- c. *Sequence and Timing.* Initially, the first timer interrupt flag register is cleared by writing 0x80 to the H11TELGI register. Then the H11TOCI register is incremented by 0x9c40 to create an interrupt at the end of 20 milliseconds. The computer operating properly timeout interrupts are prevented by assigning the values 0x55 and 0xaa to the H11COPRST register. The counter

variable "clk50" is incremented. If the variable reaches a value of 50, then the counter is reset. The down counter "dcount" is decremented by one if the counter has a value greater than zero. The counter variable "etime" is incremented at one second intervals.

- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.3.1.3 Outputs. This function has 3 counters, 1 flag and 3 hardware oriented outputs. The counters are "dcount," "pcount" and "etime" which are the two general counters and the elapsed time counter, respectively. The flag "actflag" indicates if a PMC has been activated. The register H11TFLG1 clears the interrupt flag. The register H11TOC1 is loaded to provide an interrupt rate of 50 Hz and H11COPRST is set to reject COP timer interrupts. See the table for the time keeping function output (Table 11) for additional information.

3.4.3.2 A/D Conversion Function.

3.4.3.2.1 Inputs. The function has 3 inputs. The flag which enables the A/D conversions is "adflag." The A/D conversion status byte is stored in the register H11ADCTL. And the register H11ADR3 contains the battery voltage. See the table for the A/D conversion function input (Table 12) for additional information.

3.4.3.2.2 Processing.

- a. *Intent.* This function performs the A/D conversion of the data from the voltage reference.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The A/D conversion rate is 50 Hz. The conversion begins by setting the H11ADCTL register of the 68HC11 to 0x10 which sets bit 4 of this register to 1. A conversion is complete when bit 7 of the register goes high (1). Following the conversion, the A/D value is stored in the variable "adval."
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.

- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.3.2.3 *Outputs.* The function has 2 outputs. The variable "adval" contains the voltage reference. The register H11ADCTL is used to trigger the A/D converter. See the table for the A/D conversion function output (Table 13) for additional information.

3.4.4 *Serial Receive Interrupt Function.*

3.4.4.1 *Packet Construct Function.*

3.4.4.1.1 *Inputs.* The function has 7 inputs. The variables "pin" and "pout" are input and output index pointers used in storing packet information. The variable "pcount" is a general timer used for delays. The initializing subroutine "pkclear" clears the buffer area necessary for packet construction. The serial receiver interrupt indicates the signal reception of a character by the digital RF transceiver (DRT). The register H11SCSR stores the status byte from the serial receiver. And the register H11SCDR contains the received character. See the table for the packet construction function input (Table 14) for additional information.

3.4.4.1.2 *Processing.*

- a. *Intent.* This function accepts input characters from the serial interface and attempts to construct a legal packet from the characters.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* Packet construction is accomplished by a finite state machine. Figure 4 shows the necessary flow with the variable "rstate" holding the current state of the machine.
- d. *Error Detection and Recovery.* An error routine checks the validity of the transmitted packet. The routine sums the characters in the packet to get an 8 bit checksum. Then the last two characters in the packet, which are a hex ASCII representation of the checksum, are decoded. The two checksums are compared and if they are not equal then an error occurred during packet transmission. An error results in the packet being discarded and a new packet being constructed.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.4.1.3 *Outputs.* This function has 5 outputs. The array "pkbuf" contains packet

information. The flag "status" indicates whether the packet transmission was successful. The variables "pin" and "pout" are input and output index pointers used in storing packet information. The variable "pkpt" indicates which packet is to be processed. See the table for the packet construction function output (Table 15) for additional information.

3.4.4.2 *Input Packet Processing Function.*

3.4.4.2.1 *Inputs.* This function requires 4 inputs. The array "pkbuf" is a buffer for packet storage. The variables "pin" and "pout" are input and output index pointers used in storing packet information. The variable "pkpt" indicates which packet is to be processed. See the table for the input packet processing function input (Table 16) for additional information.

3.4.4.2.2 *Processing.*

- a. *Intent.* This function decodes valid packets to determine if they are the type expected.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The packet address is decoded and stored in the structure (record) "dph." A check is made to verify the packet received is the type expected. Packet decoding continues if the packet is of type PSS and the sequence number, off-wrist flag, heart rate, motion flag and battery voltage are stored in "dph."
- d. *Error Detection and Recovery.* If the packet received is not the correct type or has the wrong address, the packet is ignored.
- e. *Restrictions or Limitations.* No restrictions or limitations pertain to this function.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.4.2.3 *Outputs.* This function has only 1 output. The output is the structure (record) "dph" which contains all of the decoded packet information. See the table for the input packet processing function output (Table 17) for additional information.

3.4.5 *Display Function.*

3.4.5.1 *Inputs.* The function has 3 inputs. The display status is stored in the register DISPLAY. The message to be displayed is located in the array "s." And "lineno" is the variable indicating the line number of the display. See the table for the display function input (Table 18) for additional information.

3.4.5.2 Processing.

- a. *Intent.* The function implements the LCD panel to display text messages for the user.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* Characters comprising the message are sent one at a time.
- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* The display is limited to 2 lines of 16 characters each.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.5.3 *Outputs.* The function has 2 outputs. The register DISPLAY stores the display status and the register DISPDAT contains the display data. See the table for the display function output (Table 19) for additional information.

3.4.6 Transmit Function.

3.4.6.1 *Inputs.* This function has 4 inputs. The variable "dcount" is used as a timer for function delays. The variable "id" contains the identification code of the selected PMC. The number of attempted retries is recorded in the variable "r." And the register I11CSR stores the status byte from the serial receiver. See the table for the transmit function input (Table 20) for additional information.

3.4.6.2 Processing.

- a. *Intent.* This function encodes and transmits valid packets to PMCs.
- b. *Parameters Affected.* No parameters are affected by this function.
- c. *Sequence and Timing.* The sequence necessary to build and transmit a valid packet is as follows. The packet characters are encoded and stored in the buffer "tpakbuf." The routine "doxss" adds the packet header ("A"), the PMCID, and the id end marker (:) to the existing packet characters. The entire string is stored in the buffer "tpakbuf." Then the routine "sendpak" calculates and encodes the packet's 8 bit checksum which is added to the packet along with a line feed and carriage return. Once the packet is built the transmitter is keyed. Following a period of 200 msec to allow the PMC receiver to stabilize, the packet is sent. The transmit key line is then turned off.

- d. *Error Detection and Recovery.* This function contains no error detection and recovery software.
- e. *Restrictions or Limitations.* The transmitter must be keyed for 200 msec before data are sent to give the modem in the receiver the time necessary to decode valid data.
- f. *Allocation of CSCI Performance Requirements.* See function block diagram (Figure 2).

3.4.6.3 *Outputs.* This function has 3 outputs. The array "tpakbuf" contains processed information to be displayed. The register H11SCDR contains the received characters and the register H11PORTA (bit 4) controls the radio power key line. See the table for the transmit function output (Table 21) for additional information.

3.5 *Adaptation Requirements.*

3.5.1 *System Environment.* This section is not applicable to this specification.

3.5.2 *System Parameters.* This section is not applicable to this specification.

3.5.3 *System Capacities.* This section is not applicable to this specification.

3.6 *Quality Factors.*

3.6.1 *Correctness Requirements.* The HHM CSCI is considered to be 100% correct.

3.6.2 *Reliability Requirements.* The software has been developed to consistently provide meaningful results regardless of the input signal.

3.6.3 *Efficiency Requirements.* The CSCI uses the computer resources in a manner which accomplishes status cycle evaluation and reporting in less than 15 seconds.

3.6.4 *Integrity Requirements.* No control against unauthorized access to operations and data exists in this CSCI.

3.6.5 *Usability Requirements.* The CSCI design assures the HHM is as simple to use as possible. Once the HHM unit is activated, the CSCI provides all necessary operation instructions on the LCD display.

3.6.6 *Maintainability Requirements.* The maximum effort necessary to locate and fix an error should not exceed 2 human work weeks.

3.6.7 Testability Requirements. Testing of the CSCI should range from 1 day to 1 week depending on the data set selected. Sections of the CSCI are difficult to independently test due to their close relationship to the hardware.

3.6.8 Flexibility Requirements. Minimal time will be required for enhancements due to the modular nature of the HHM CSCI.

3.6.9 Portability Requirements. The software is portable due to the facts it is primarily written in C and it is modular. However, since the software has been developed for a particular microprocessor with built-in interfaces, complications may result when attempting to re-port the software.

3.6.10 Reusability Requirements. The HHM CSCI will be reusable in other applications with minimal effort since the software is written in the high level language C and is modular.

3.6.11 Interoperability Requirements. The CSCI is not compatible with any existing systems except for the personal monitor communicator (PMC) and the prototype multimonitor.

3.6.12 Additional Quality Factor Requirements. This section is not applicable to this specification.

3.7 CSCI Support.

3.7.1 Facility Requirements. Laboratory workspace suitable for the equipment and personnel listed below is necessary. The workspace will need to have standard 120 volt AC power, temperature control for the room, a computer table, and a workbench.

3.7.2 Equipment Requirements. The following equipment is necessary:

- a. IBM PC compatible computer with a serial port to talk to the emulator (recommend a 386 system with a hard disk drive)
- b. Motorola HDS-300 Development System with a 68HC11 emulator pod
- c. Wire-wrap prototype of PMC and HHM hardware with a digital to analog converter for debugging and testing software during development
- d. 100 MHz bandwidth oscilloscope to monitor software functions
- e. Five volt DC power supply for prototype circuit
- f. PROM programmer to attach to the IBM PC compatible computer for programming 27C64 and 27C256 EPROMS

7.3 Software Requirements. The following software is necessary:

- a. INTROL-C/68HC11 C cross compiler for the Motorola 68HC11 microprocessor which runs on the IBM PC compatible computer
- b. MS-DOS for the IBM PC compatible computer
- c. Text editor
- d. Copies of the PMC and HHM source code
- e. File transfer program for PC to HDS-300 communication
- f. C compiler for the IBM PC compatible computer to compile and test the software described in item 6 (above)

7.4 Personnel Requirements. The personnel maintaining and developing the PMC and HHM software need knowledge in the following areas:

- a. C programming
- b. stand-alone microprocessor programming (MC68HC11 in particular)
- c. real time programming (interrupt handling, etc.)

3 Traceability. This section is not applicable to this specification.

QUALIFICATION REQUIREMENTS

1 General Qualification Requirements. Testing of the CSCI on an independent basis has not been performed. See documentation concerning testing in the Interface Requirement Specification (A006).

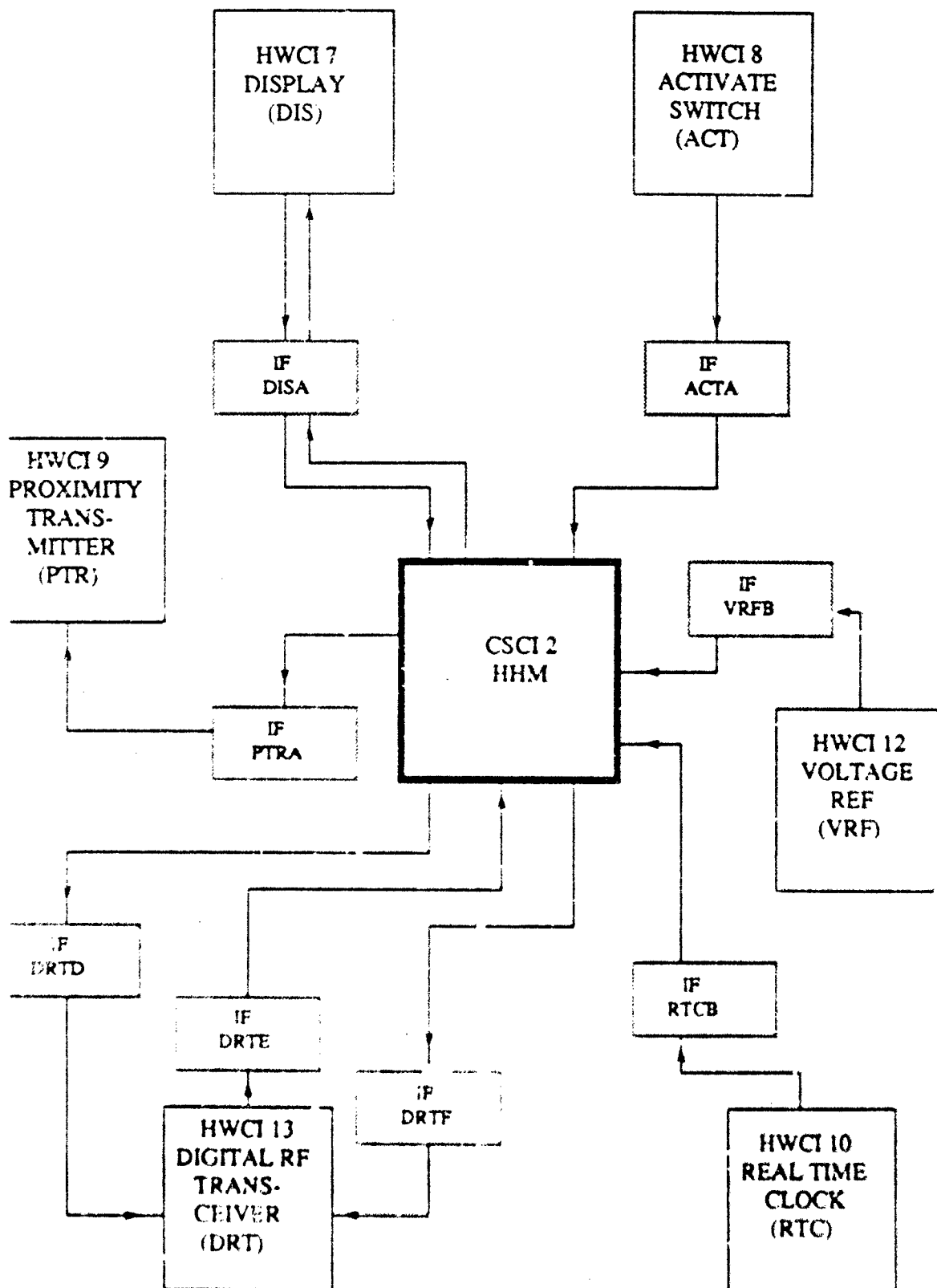
2 Special Qualification Requirements. This section is not applicable to this specification.

PREPARATION FOR DELIVERY

1 Preparation for Delivery. The CSCI software will be delivered in object code form on an EPROM socketed into the HHM unit. A hardcopy of the source code can be found in the appendix which accompanies this documentation.

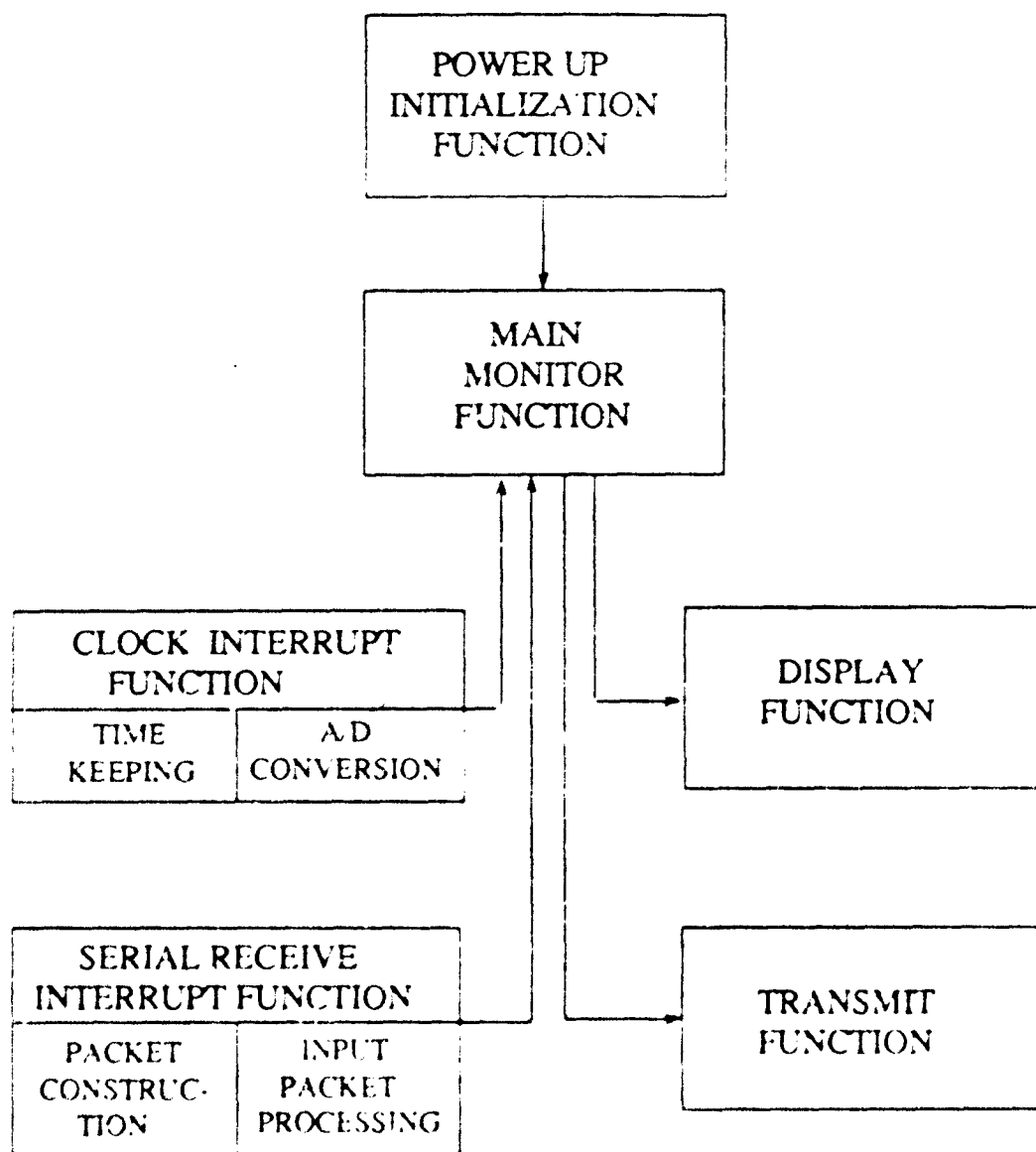
NOTES

1 Acronyms. See acronym summary table (Table 22).



NOTE:
- INTERFACE

FIGURE 1. Interface block diagram.

FIGURE 2. *Function block diagram.*

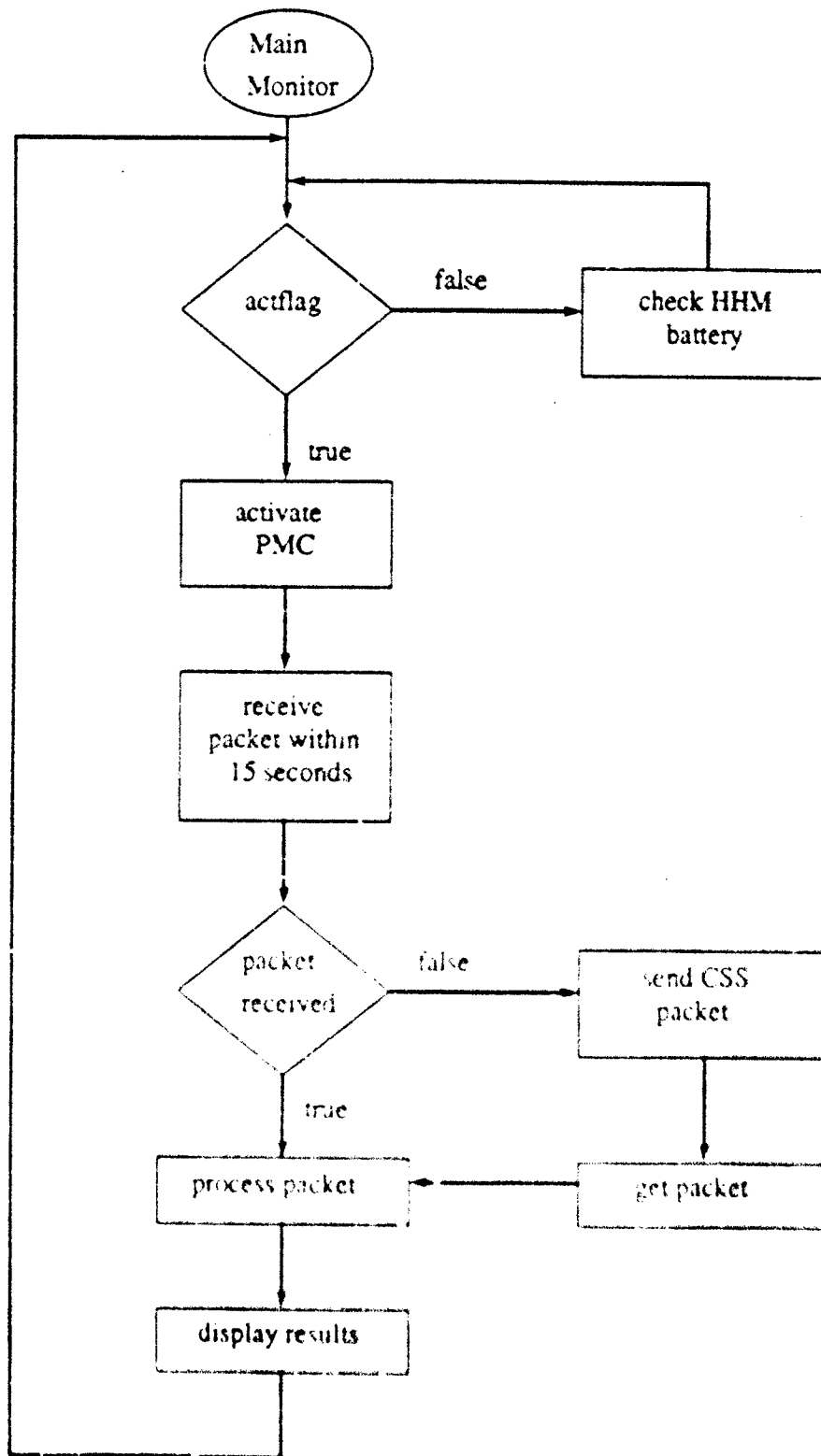


FIGURE 3. Function main monitor flowchart.

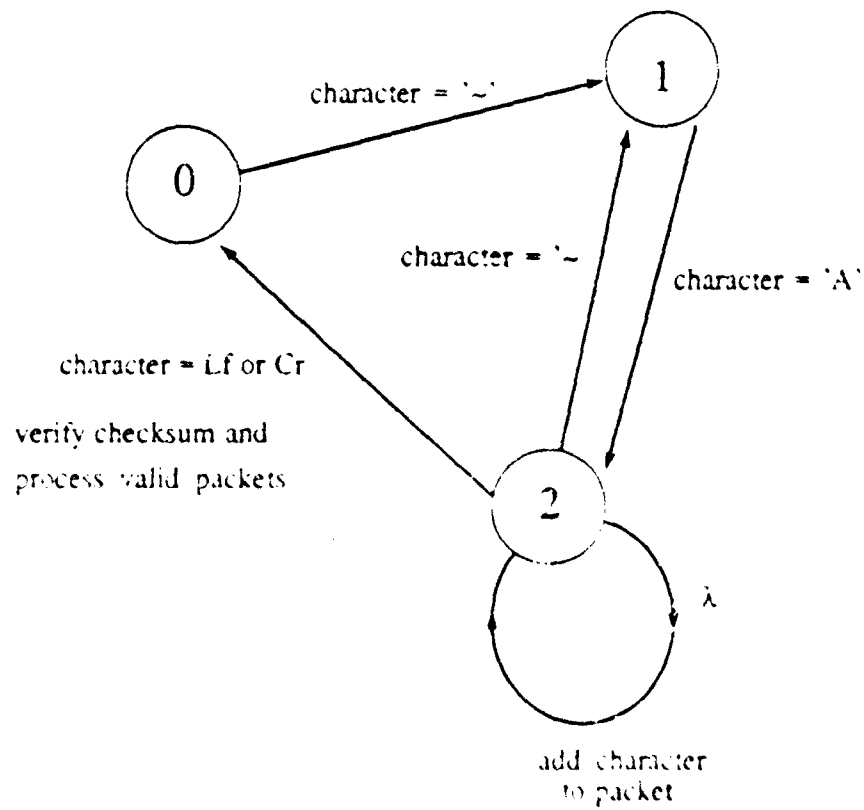


FIGURE 4. *Function packet construction finite state machine.*

TABLE 1. Interface identification/documentation.

INTERFACE NAME	INTERFACING ITEM		INTERFACE DOCUMENT NAME	INTERFACE DOCUMENT PARAGRAPH
	TITLE	CI NUMBER		
DISP	DISPLAY	7	SRS for HHM -- DISP interface	3.3.3.2.3
ACTS	ACTIVATE SWITCH	8	SRS for HHM -- ACTS interface	3.3.3.2.4
PTRM	PROXIMITY TRANSMITTER	9	SRS for HHM -- PTRM interface	3.3.3.2.5
RTCB	REAL TIME CLOCK	10	SRS for HHM -- RTC interface	3.3.3.2.6
VRFB	VOLTAGE REF	12	SRS for HHM -- VRFB interface	3.3.3.2.1
DRTD	DIGITAL RF TRANSCEIVER	13	SRS for HHM -- DRT interface	3.3.3.2.2
DRTE	DIGITAL RF TRANSCEIVER	13	SRS for HHM -- DRT interface	3.3.3.2.2
DRTF	DIGITAL RF TRANSCEIVER	13	SRS for HHM -- DRT interface	3.3.3.2.2

TABLE 2. *Interface summary.*

INTERFACE NAME	INFORMATION DESCRIPTION	INITIATION CONDITION	EXPECTED RESPONSE
VRFB	Analog reference voltage HWCI 12-to-CSCI 2	HHM activated	Check A/D voltage reference
DRTD	Transmit data serial communication CSCI 2-to-HWCI 13	HHM sending data packet	Data packet sent by RF transceiver
DRTE	Receive data serial communication HWCI 13-to-CSCI 2	DRT receiving data packet	Data packet processed
DRTF	Digital RF transceiver power CSCI 2-to-HWCI 13	HHM activated	DRT activated
DISP	HHM display CSCI 2-to-HWCI 7 HWCI 7-to-CSCI 2	HHM receives activate or message	Display message
ACTS	Activate system HWCI 8-to-CSCI 2	Activate switch pressed	Activates HHM
PTRM	Proximity data CSCI 2-to-HWCI 9	HHM activated	Proximity receiver activated
RTCB	Real time clock interrupt HWCI 10-to-CSCI 2	HHM powered up or PMC activated	Internal interrupt at 50 HZ

TABLE 3. *Request for soldier status (CSS) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	CSS	packet type
1 byte*	ASCII		space (blank)
1 byte*	ASCII	I	signals CSCI 1 response
2 bytes	ASCII	hex digits	packet checksum
2 bytes	ASCII	LfCr	packet trailer

* optional

TABLE 4. *PMC positive acknowledge (PPA) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	PPA	packet type
2 bytes	ASCII	hex digits	packet checksum
2 bytes	ASCII	LfCr	packet trailer

TABLE 5. *PMC soldier status (PSS) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	PSS	packet type
1 byte	ASCII		space (blank)
1 to 5 bytes	ASCII	1 to 5 digits	evaluation count
1 byte	ASCII		space (blank)
1 to 3 bytes	ASCII	1 to 3 digits	off-wrist level
1 byte	ASCII		space (blank)
2 to 3 bytes	ASCII	2 to 3 digits	heart rate
1 byte	ASCII		space (blank)
1 to 4 bytes	ASCII	1 to 4 digits	motion count
1 byte	ASCII		space (blank)
1 to 3 bytes	ASCII	1 to 3 digits	A/D reference level
1 byte	ASCII		space (blank)
2 bytes	ASCII	LfCr	packet trailer

TABLE 6. *Function power up initialization input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Power up reset	N/A	Once when device is turned on	N/A	CSCI 2

TABLE 7. *Function power up initialization output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	A/D conversion enable flag (adflag)	N/A	N/A	N/A	Function A/D Conversion
2	Activate flag (actflag)	N/A	N/A	N/A	Function Main Monitor or Time Keeping
3	Buffer input index (pin)	N/A	N/A	N/A	Function Power Up
4	Buffer output index (pout)	N/A	N/A	N/A	Function Power Up
5	Initialization subroutine (pkclear)	N/A	N/A	N/A	Function Power Up
6	Display operating instructions	N/A	N/A	N/A	Function Display
7	Baud rate register (H11BAUD)	N/A	N/A	N/A	HWCI 6
8	Port D direction register (H11DDRD)	N/A	N/A	N/A	HWCI 6

TABLE 7, continued
Function power up initialization output.

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
9	A/D conversion (H11OPTION)	N/A	N/A	N/A	HWCI 3
10	Key line (H11PORTA bit 4)	N/A	N/A	N/A	HWCI 6
11	Serial interface register 1 (H11SCCR1)	N/A	N/A	N/A	HWCI 6
12	Serial interface interrupt (H11SCCR2)	N/A	N/A	N/A	HWCI 6
13	Output compare 1 interrupt (H11TMSK1)	N/A	N/A	N/A	HWCI 11
14	Output compare 1 register (H11TOC1)	N/A	N/A	N/A	HWCI 11

TABLE 8. *Function main monitor input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Battery voltage reference (adval)	N/A	50 Hz	No	Function A/D Conversion
2	PMC identification (id)	N/A	N/A	N/A	Function Input Packet Processing
3	Receive packet status (status)	N/A	N/A	Yes	Function Packet Construction
4	General counter (dcount)	1/50 Seconds	N/A	No	Function Time Keeping
5	Elapsed time counter (etime)	Seconds	1/Second	No	Function Time Keeping
6	Activate flag (actflag)	N/A	N/A	N/A	Function Power Up or Time Keeping
7	Decoded packet buffer (dpb)	N/A	N/A	N/A	Function Input Packet Processing

TABLE 9. *Function main monitor output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Retry number (retries)	N/A	N/A	N/A	Function Transmit
2	PMC identification (id)	N/A	N/A	N/A	Function Transmit
3	Elapsed time counter (etime)	Seconds	N/A	N/A	Function Time Keeping
4	General timer (dcount)	1/50 Seconds	N/A	N/A	Function Time Keeping
5	Activate flag (actflag)	N/A	N/A	N/A	Function Time Keeping
6	Message for display	N/A	N/A	N/A	Function Display

TABLE 10. *Function time keeping input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Clock interrupt	N/A	50 Hz	N/A	HWCI 10
2	General counter (dcount)	1/50 Seconds	N/A	Yes	Function Main Monitor
3	Elapsed time counter (etime)	Seconds	N/A	Yes	CSCI 2
4	Activate flag (actflag)	N/A	N/A	N/A	Function Power Up or Main Monitor

TABLE 11. *Function time keeping output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	General counter (dcount)	1/50 Seconds	N/A	N/A	Function Main Monitor
2	General counter (pcount)	1/50 Seconds	N/A	N/A	Function Packet Construction
3	Elapsed time counter (etime)	Seconds	N/A	N/A	Function Main Monitor
4	Activate flag (actflag)	N/A	N/A	N/A	Function Main Monitor
5	Interrupt flag (HIITFLG1)	N/A	N/A	N/A	HWCI 10
6	Timing register (HIITOC1)	N/A	N/A	N/A	HWCI 10
7	Timer interrupt (HIICOPRST)	N/A	N/A	N/A	HWCI 10

TABLE 12. *Function A/D conversion input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	A/D conversion enable flag (adflag)	N/A	N/A	N/A	Function Power Up
2	A/D conversion status byte (H11ADCTL)	N/A	N/A	N/A	HWCI 12
3	Battery voltage (H11ADR3)	N/A	25 Hz	No	HWCI 12

TABLE 13. *Function A/D conversion output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Battery voltage (adval)	N/A	N/A	No	Function Main Monitor
2	A/D conversion control byte (H11ADCTL)	N/A	N/A	N/A	HWCI 12

TABLE 14. *Function packet construction input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Buffer input index (pin)	N/A	N/A	N/A	Function Power Up
2	Buffer output index (pout)	N/A	N/A	N/A	Function Power Up
3	General timer (pcount)	1/50 Seconds	N/A	N/A	Function Time Keeping
4	Initialization subroutine (pkclear)	N/A	N/A	No	Function Power Up
5	Serial receiver interrupt	N/A	N/A	N/A	HWCI 13
6	Serial receiver status byte (H11SCSR)	N/A	N/A	N/A	HWCI 13
7	Character register (H11SCDR)	N/A	N/A	N/A	HWCI 13

TABLE 15. *Function packet construction output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Packet storage (pkbuf)	N/A	N/A	Yes	Function Input Packet Processing
2	Receive packet status (status)	N/A	N/A	N/A	Function Main Monitor
3	Buffer input index (pin)	N/A	N/A	N/A	Function Input Packet Processing
4	Buffer output index (pout)	N/A	N/A	N/A	Function Input Packet Processing
5	Packet buffer index (pkpt)	N/A	N/A	Yes	Function Input Packet Processing

TABLE 16. *Function input packet processing input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Packet storage buffer (pkbuf)	N/A	N/A	N/A	Function Packet Construction
2	Buffer input index (pin)	N/A	N/A	N/A	Function Packet Construction
3	Buffer output index (pout)	N/A	N/A	N/A	Function Packet Construction
4	Packet buffer index (pkpt)	N/A	N/A	N/A	Function Packet Construction

TABLE 17. *Function input packet processing output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Decoded packet buffer (dpb)	N/A	N/A	N/A	Function Main Monitor or Display

TABLE 18. *Function display input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	Display status register (DISPLAY)	N/A	N/A	N/A	HWCI 7
2	Message (s)	N/A	N/A	N/A	Function Power Up or Main Monitor or Input Packet Processing
3	Display line number (lineno)	N/A	N/A	N/A	Function Power Up or Main Monitor or Input Packet Processing
4	Decoded packet buffer (dpb)	N/A	N/A	N/A	Function Input Packet Processing

TABLE 19. *Function display output.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINATION
1	Display status register (DISPLAY)	N/A	N/A	N/A	HWCI 7
2	Display data register (DISDAT)	N/A	N/A	N/A	HWCI 7

TABLE 20. *Function transmit input.*

ITEM	DESCRIPTION	UNITS OF MEASURE	FREQUENCY OF ARRIVAL	LEGALITY CHECK	SOURCE
1	General timer (dcount)	1/50 Seconds	N/A	N/A	Function Time Keeping
2	PMC identification (id)	N/A	N/A	N/A	Function Main Monitor
3	Retry number (r)	N/A	N/A	N/A	Function Main Monitor
4	Serial receiver status byte (H11SCSR)	N/A	N/A	N/A	HWCI 13

TABLE 21. *Function transmit output.*

ITEM	DESCRIP- TION	UNITS OF MEASURE	FREQUENCY OF DEPARTURE	LEGALITY CHECK	DESTINA- TION
1	Transmit packet (tpakbuf)	N/A	N/A	Yes	HWCI 13
2	Transmit packet (H11SCDR)	N/A	N/A	Yes	HWCI 13
3	Transmit key line (H11PORTA bit 4)	N/A	N/A	N/A	HWCI 13

TABLE 22. *Acronym summary.*

ACRONYM	MEANING
ACT	Activate switch
ACTA	Activate switch to CSCI 2 interface
CSS	Request for soldier status
DIS	Display
DISA	Display/CSCI 2 interface
DRT	Digital RF transceiver
DRTD	CSCI 2 to DRT interface
DRTE	DRT to CSCI 2 interface
DRTF	CSCI 2 to DRT interface
HHM	Hand held monitor
PMC	Personal monitor communicator
PPA	PMC positive acknowledge
PTR	Proximity transmitter
PTRA	CSCI 2 to PTR interface
PSS	PMC soldier status
RTC	Real time clock
RTCB	RTC to CSCI 2 interface
VRF	Voltage Reference
VRFB	VRF to CSCI 2 interface

APPENDIX I.

10. HHM SOURCE CODE

```

/* HHM program */
/* written by Jim Jones, George Graber */
/* Purdue University */
/* 1986-1989 */

#define HHMID 4 /* HHM ID number, should be unique */
#include <stdio.h>
#include <hllreg.h>
#include <bmec.h>
#include <terminal.h>

/* ASCII constants */
#define LF 10
#define CR 13

#define MAXRETRIES 10 /* maximum number of retries */
#define EVALTIME 15 /* time for PMC to complete an evaluation */

#define PPA 1 /* packet types */
#define PSS 2

#define PBUFSIZE 8 /* input packet buffer size */
#define PMASK 0x07

#define PASSIVE 0 /* HHM monitor mode */
#define ACTIVE 1

/* structure used to set up serial interface */
struct ioctl {ttyb;

char clk50; /* 50 HZ counter for real time clock */
int dcount; /* 50 HZ down counter for general purpose timing */
int pcount;
int etime; /* 50 HZ elapsed time counter */

char tpakbuf[200]; /* buffer to construct packets for transmission */
char rpakbuf[100];
char pkbuf[PBUFSIZE+2*(100)]; /* buffer to store incoming packets */

struct { /* buffer to store decoded packets */

```

```

char pk_type;
int pk_id;
int pk_seq;
int pk_hr;
int pk_offw;
int pk_batt;
int pk_mot;
}dpb[PBUFSIZE+2];

int pin,pout;      /* input and output indexes for packet buffer */
char *pakptr;      /* pointer to receive packet buffer */
int rstate;        /* state of receive packet finite state machine */
int chksum;         /* check sum calculated on received packets */
int rchksum;        /* check sum included in received packets */

char line[80];      /* buffer for display messages */

/* soldier status values returned from a PMC */
int evnum;          /* evaluation number */
int offw;           /* PMC off wrist flag */
int hrate;          /* heart rate */
int batt;           /* battery voltage */
int mocount;        /* motion counter */
int id;             /* PMC ID */

char acflag;        /* ACTIVATE Pushbutton flag */

int adflag;         /* enable A/D conversion flag */
int adval;          /* value from A/D conversion */

int retries;        /* number of retries for communication */
int restart;        /* flags for main monitor loop */
int done;
int mode;           /* monitor mode (ACTIVE or PASSIVE) */

/* routine declarations */
void int_setup();
__mod2__ void OCINT();
__mod2__ void INTINT();
__mod2__ void SCIINT();

main(){

    /* turn of transmitter off */
    key(0);
    pin = pout = 0;
    pkclear();

```

```

/* set up interrupts */
int_setup();

/* set up serial port */
tty_setup();

/* initialise LCD display */
init_disp();

/* Power up message */
message(0," Hand Held");
message(1," Monitor (HBM)");
sleep(2);
instruct();

/* turn on A/D converter */
adon();

/* main monitor loop */
while(1){

    /* ACTIVATE button pressed */
    if(actflag){
        activate();
        mode = ACTIVE;
        while(actflag == 0){
            retries = 0;
            etime = 0;
            restart = 0;
            while(etime < EVALTIME*50){
                if(actflag)break;
                while(pout != pin){
                    if(dpb'pout'.k_id == id){
                        if(dpb'pout'.pk.type == PPA){
                            /* found PPA packet */
                            restart = 1;
                        }
                        if(dpb'pout'.pk.type == PSS){
                            /* found PSS packet */
                            restart = 1;
                            /* display status results */
                            dopes(pout);
                        }
                    }
                }
                pout = (pout + 1) & PMASK;
            }
        }
    }
}

```

```

if(actflag)break;
if(restart){
    mode = PASSIVE;
}
else{
    retries = 0;
    done = 0;
    do{
        sprintf(line,"PMC Query ");
        message(0,"Attempting PMC");
        message(1,"Communication");
        docss(id,retries);
        etime = 0;
        do{
            while(pout != pin){
                if(dpb[pout].pk_id == id){
                    if(dpb[pout].pk_type == PPA){
                        mode = PASSIVE;
                        done = 1;
                    }
                    if(dpb[pout].pk_type == PSS){
                        done = 1;
                        dopss(pout);
                    }
                }
                pout = (pout + 1) & PMASK;
            }
            }while(etime < rtime(retries) && !done && !actflag);
            retries++;
        }while(retries < MAXRETRIES && !done && !actflag);
        if(!done){
            message(0," Lost");
            message(0,"Communication");
            sleep(2);
            instruct();
            break;
        }
    }
}
if(mode == PASSIVE)sleep(2);
/* check hlm battery */
if(adval >= 68){
    sleep(2);
    sprintf(line,"Low HLM Battery");
    message(1,line);
}
}

```

```

    }
}
/* attempt to active a PMC */
activate(){
    retries = 0;

    /* display activate message */
    sprintf(line,"Attempting");
    message(0,line);
    message(1,"PMC Activation");
    do{
        /* clear packet buffer */
        pout = pin;
        pkclear();
        /* pulse proximity transmitter */
        prox();
        id = 0;
        /* check for PMC response */
        etime = 0;
        do{
            if(pout != pin){
                if(dpb[pout].pk_type == PPA){
                    id = dpb[pout].pk_id;
                }
                pout = (pout + 1) & PMASK;
            }
        }while(etime < rtime(retries) && id == 0);
        retries++;
    }while(id == 0 && retries < MAXRETRIES);

    actflag = 0;

    /* if id == 0 then activation was unsuccessful */
    if(id == 0){
        message(0,"ERROR--Repeat");
        message(1,"Activate Steps");
        sleep(2);
        instruct();
    }
    /* else PMC was activated */
    else {
        actflag = 0;
        sprintf(line,"PMC ACTIVATED",id);
        message(0,line);
        message(1,"");
        dcount = 25;
        while(dcount);
    }
}

```



```

    }
}
/* output operating instructions to the display */
instruct(){
    sprintf(line,"Put HHM Near PMC");
    message(0,line);
    sprintf(line,"Press ACTIVATE");
    message(1,line);
}

/* calculate a variable retry time based on retry number */
rttime(r)int r;{
    int i;
    int rt;
    switch(r){
        case 1:
        case 6:
            rt = 0;
            break;
        case 2:
        case 7:
            i = clk50;
            rt = 8 * i;
            break;
        case 3:
        case 8:
            rt = (HHMID & 7) * 50;
            break;
        case 4:
        case 9:
            rt = (adval & 7) * 50;
            break;
        case 5:
        case 10:
            rt = id * 50;
            break;
        default:
            rt = 100;
            break;
    }
    return(rt+100);
}

/* pause for i seconds */
sleep(i)int i;{
    etime = 0;
    while(etime < i*50);
}

```

```

    }

/* activate proximity transmitter */
prox(){
    H11PORTA |= 0x20;
    dcount = 5;
    while(dcount);
    H11PORTA &= 0xdf;
}

/* process and display results of PSS packet */
dopss(pkpt){
    int pkpt;
    evnum = dpb[pkpt].pk_seq;
    offw = dpb[pkpt].pk_offw;
    hrate = dpb[pkpt].pk_hr;
    mocount = dpb[pkpt].pk_mot;
    batt = dpb[pkpt].pk_batt;

    message(1,"");
    if(offw > 190){
        sprintf(line,"Check Wrist");
        message(0,line);
        sprintf(line,"Strap");
        message(1,line);
    }
    else{
        if(hrate >= 50){
            sprintf(line,"Heart Rate %3d",hrate);
            message(0,line);
        }
        else if(hrate < 50 && hrate > 0 && mocount == 0){
            sprintf(line,"Heart Rate %3d",hrate);
            message(0,line);
        }
        else {
            sprintf(line,"Heart Rate ?");
            message(0,line);
        }
        strcpy(line,"");
        if((hrate == 0) || (hrate == -4))sprintf(line,"Low Pulse");
        if(mocount)sprintf(line,"Motion");
        message(1,line);
    }
    if(batt >= 64){
        sleep(2);
        sprintf(line,"Low PMC Battery");
    }
}

```

```

        message(1,line);
    }
    return(1);
}

/* transmit CSS packet */
docss(id,r) int id,r; {
    char *p;
    if(mode == PASSIVE){
        if(r > 5) mode = ACTIVE;
    }
    if(mode == ACTIVE){
        p = &tpakbuf[0];
        if(r) printf(p, "A%d:CSS I", id);
        else sprintf(p, "A%d:CSS", id);
        sendpak(p);
    }
}

/* send a packet through the radio transmitter */
sendpak(pk) char *pk; {
    int i;
    char *p;
    p = pk;
    for(i = 0; *p; p++) i = i + *p;
    i = i & 0xff;
    *p = (i >> 4) + '0';
    if(*p > '9') *p = *p + 7;
    p++;
    *p = (i & 0xf) + '0';
    if(*p > '9') *p = *p + 7;
    p++;
    *p = LF;
    p++;
    *p = CR;
    p++;
    *p = 0;
    key(1);
    dcount = 10;
    while(dcount);
    p = pk;
    pkputc('');
    pkputc('X');
    while(*p){
        pkputc(*p);
        p++;
    }
}

```

```

    pkputc(CR);
    pkputc(CR);
    key(0);
}

/* output a character to the radio */
pkputc(c) char c;{
    while((H11SCSR & 0x80) == 0);
    H11SCDR = c;
}

/* get a character from the radio */
pkgetc(){
    while((H11SCSR & 0x20) == 0);
    return(H11SCDR);
}

/* clear the packet buffer */
pkclear(){
    pakptr = pkbuf[pin];
    chksum = 0;
    rstate = 0;
}

/* SCIINT is entered when a character is received */
/* SCIINT constructs packets as they are received */
__mod2__ void SCIINT(){
    int c;
    int pt;
    char *s;
    c = pkgetc();
    /* pkputc(c); */
    switch(rstate){
        case 0:
            if(c == '\n'){
                *pakptr++ = c;
                chksum += c;
                rstate = 1;
            }
            break;
        case 1:
            if(c == '\n') break;
            if(c == 'A'){
                *pakptr++ = c;
                chksum += c;
                rstate = 2;
                pcount = 50;
            }
    }
}

```

```

    }
    else pkclear();
    break;
case 2:
    if(c == '\n'){
        pkclear();
        *pakptr++ = c;
        chksum = c;
        rstate = 1;
    }
    else if((c == '0') || (c == ',')){
        *pakptr = 0;
        s = pakptr;
        s--; s--;
        rchksum = xtoi(s);
        chksum -= *s++;
        chksum -= *s;
        chksum &= 0xff;
        if(chksum == rchksum){
            pt = pin;
            pin = (pin + 1) & PMASK;
            pkclear();
            dopacket(pt);
        }
        else{
            pkclear();
            asm("    CLI");
        }
    }
    else {
        if((pakptr >= &pkbuf[pin][49]) || (pcount == 0))pkclear();
        else{
            *pakptr++ = c;
            chksum += c;
        }
    }
    break;
}
}

```

/* dpacket decodes received packets */

dopacket(pkpt)int pkpt;{

int n;

char *num[10];

asm(" CLI");

/* process pkbuf[pkpt] */

s = pkbuf[pkpt];

```

s++; s++;
for(n=0; isdigit(s[n]) && n < 9; u++) num[n] = s[n];
num[n] = ' ';
dpb[pkpt].pk_id = atoi(num);
s = s + n + 1;
if(strncmp(s, "PPA", 3) == 0){
    dpb[pkpt].pk_type = PPA;
}
else if(strncmp(s, "PSS", 3) == 0){
    dpb[pkpt].pk_type = PSS;
    for(n=0; pkbuf[pkpt][n] != ':'; n++);
    n = n + 4;
    s = &pkbuf[pkpt][n];
    sscanf(s, "%d %d %d %d %d %d", &dpb[pkpt].pk_seq, &dpb[pkpt].pk_offw,
        &dpb[pkpt].pk_hr, &dpb[pkpt].pk_mot, &dpb[pkpt].pk_batt);
}
}

__mod2__ void INTINT(){

/* key the transmitter */
key(i) int i;{
    if(i == 0) H11PORTA |= 0x08;
    else H11PORTA &= 0xF7;
}

/* service the OC1 interrupt (real time clock, 50HZ) */
__mod2__ void OC1INT(){
    H11TFLG1 = 0x80;
    H11TOC1 += 0x9c40;
    H11COPRST = 0x55;
    H11COPRST = 0xaa;
    if(((H11PORTA & 0x02) == 0) && !actflag) actflag = 1;
    if(actflag){
        H11ADCTL = 0x10;
        while(H11ADCTL & 0x80 == 0);
        adval = H11ADR1;
    }
    clk50++;
    etime++;
    if(dcount) dcount--;
    if(pcount) pcount--;
    if(clk50 >= 50){
        clk50 = 0;
    }
}

```

```

__mod2__ void COPINT(){
    message(0,"COP RESET!");
}

```

```

/* set up interrupt service routines */
void int_setup(){

```

```

    __mod2__ void OC1INT();
    __mod2__ void INTINT();
    __mod2__ void COPINT();
    __mod2__ void SCIINT();
    void VECTOR();
    dcount = 0;
    pcount = 0;
    VECTOR(OC1INT,9);
    VECTOR(INTINT,14);
    VECTOR(COPINT,17);
    VECTOR(COPINT,18);
    VECTOR(SCIINT,0);
    H11TOC1 = 0;
    H11TMSK1 |= 0x80;
}

```

```

/* set up serial interface */

```

```

tty_setup(){
    chksum = 0;
    rchksum = 0;
    rstate = 0;
    H11SCCR1 = 0;
    H11SCCR2 = 0x2C;
    H11BAUD = 0x35;
    H11DDRD = 2;
    H11PORTD = 0;
}

```

```

/* is serial interface ready */

```

```

com_rdy(){
    if(H11SCSR & 0x20)return(1);
    else return(0);
}

```

```

adon(){

```

```

    H11OPTION |= 0x80; /* power up a/d */
    adflag = 1;
}

```

```

#include "hex.c"

```

```

#include "display.c"

```

```
/* hex.c program */
```

```
/* convert hex ascii string to integer */
```

```
atoi(s)char *s;{
    int i;
    i=ctoh(s[0]);
    i=(i<<4) | ctoh(s[1]);
    return(i);
}
```

```
/* convert hex ascii digit to integer */
```

```
ctoh(c1)
int c1;
{
    if(c1>96)c1=c1-32;
    c1=c1-48;
    if(c1>9)c1=c1-7;
    return(c1);
}
```

```
/* convert integer to hex ascii character */
```

```
char htoc(i) int i;{
    char h;
    h = i + '0';
    if(h > '9')h += 7;
    return(h);
}
```



```

/* display.c program */

#define DWAIT {while(DISPLAY & 0x80);}
#define BWAIT {dcount = 2; while(dcount);}

/* initialize the LCD display */
init_disp(){
    BWAIT;
    dctrl(0x38);
    BWAIT;
    dctrl(0x38);
    BWAIT;
    dctrl(0x38);
    BWAIT;
    dctrl(0x38);
    dctrl(0x0c);
    dctrl(0x06);
    dctrl(1);
}

/* output control byte to the LCD
dctrl(v)int v:{
    DWAIT;
    DISPLAY = v;
}

/* output a character to the LCD */
dputc(c)int c:{
    DWAIT;
    DISPDAT = c;
}

/* output a string to the LCD */
dputs(s) char *s;{
    while(*s){dputc(*s); s++;}
}

/* output a message to the LCD */
message(i,s) int i; char *s;{
    if(i == 0)i = 0x80;
    else i = 0xc0;
    dctrl(i);
    dputc(" ");
    dctrl(i);
    dputs(s);
}

```

INTERFACE REQUIREMENTS SPECIFICATION
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A006

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

CONTENTS

Paragraph	Page
1. SCOPE.....	4
1.1 Identification	4
1.2 Purpose	4
1.3 Introduction	4
2. APPLICABLE DOCUMENTS.....	4
2.1 Government Documents.....	4
2.2 Non-Government Documents	4
3. REQUIREMENTS.....	5
3.1 Interface Relationships.....	5
3.2 Interface Identification and Documentation	5
3.3 Detailed Interface Requirements.....	5
3.3.1 CSCI-to-CSCI Interface Requirements.....	5
3.3.2 CSCI-to-HWCI or Critical Item Requirements.....	5
3.3.2.1 Proximity Transmitter/Receiver Interface	5
3.3.2.2 Digital RF Transceiver Interfaces	6
4. QUALIFICATION REQUIREMENTS	7
4.1 General Qualification Requirements	7
4.2 Special Qualification Requirements	7
6. NOTES.....	7
6.1 Notes.....	7

FIGURES

Figure	Page
1. Interface block diagram	8

CONTENTS - continued

TABLES

Table	Page
1. Interface identification/documentation	9
2. Interface summary	10
3. Request for soldier status (CSS) format	11
4. PMC positive acknowledge (PPA) format	12
5. PMC soldier status (PSS) format	13
6. Acronym summary	14

APPENDIX

Appendix	Page
Appendix I	
10 Interface Test Summary,	15

1. SCOPE

1.1 *Identification.* This Interface Requirements Specification establishes the requirements for the interfaces identified as the HHM-to-PMC proximity transmission (PRTA) and the PMC-HHM RF transceiver (DRTG and DRTH) for the CSCIs identified as Personal Monitor Communicator Software, PMC-CSCI.1, and Hand Held Monitor Software, HHM-CSCI.2, in the Personal Monitor and Communicator, PMC-SYS.1, System.

1.2 *Purpose.* The three interfaces are necessary for accurate communications between the PMC and the HHM.

1.3 *Introduction.* This document provides detailed and complete specifications of the communication interfaces between the personal monitor (PMC) and the hand held monitor (HHM) which compose the Personal Monitor and Communicator, PMC-SYS.1, System.

2. APPLICABLE DOCUMENTS

2.1 *Government documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

SPECIFICATIONS

Military

MIL-S-83490 Specifications, Types and Forms

STANDARDS

Military

DOD-STD-2167 Defense System Software Development

MIL-STD-483A Configuration Management Practices For
Systems, Equipment, Munitions, and Computer
Programs

MIL-STD-490A Specifications Practices

2.2 *Non-Government documents.* The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

OTHER PUBLICATIONS

Oppenheim, A.V., Schaffer, R.W.
Digital Signal Processing
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975

Kernighan, B.W., Ritchie, D.M.
The C Programming Language
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1978

Introl C MS-DOS Host Guide Version 1.0
Introl Corporation, Milwaukee, WI, 1987

Motorola Semiconductor Technical Data
Advance Information for the MC68HC11A8
HCMOS Single-Chip Microcomputer
Order number MC68HC11A8/D

Programmer's Reference Manual
M68HC11 HCMOS Single-Chip Microcomputer
Order number M68HC11PM/AD

3. REQUIREMENTS

3.1 *Interface Relationships.* See interface identification figure (Figure 1).

3.2 *Interface Identification and Documentation.* See interface identification table (Table 1).

3.3 *Detailed Interface Requirements.*

3.3.1 *CSCI-to-CSCI Interface Requirements.* This section is not applicable to this specification.

3.3.2 *CSCI-to-HWCI or Critical Item Requirements.*

3.3.2.1 *Proximity Transmitter/Receiver Interface.*

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The transmitted signal is a 125 KHz electric-field proximity signal with an upper transmission range of 4 inches.

- c. *Memory Buffer And Location.* This section is not applicable to this interface.
- d. *Transfer Protocol.* This section is not applicable to this interface.
- e. *Initiation Condition.* See interface table (Table 2).
- f. *Priority Level.* This interface has the highest priority in the system.
- g. *Expected Response.* Table 2 shows the expected response to the PMC activation. If no response is detected, the proximity signal is re-transmitted after a variable-time delay.

3.3.2.2 Digital RF Transceiver Interfaces.

- a. *Signal Direction.* See interface table (Table 2).
- b. *Signal Format.* The HHM-to-PMC interface (IF DRTG) can transmit one message. The format of the request for the soldier status message (CSS) is shown in Table 3. The PMC-to-HHM interface (IF DRTH) is capable of sending two messages. The formats for the transmission of the PMC positive acknowledge (PPA) and the PMC soldier status (PSS) are shown in Tables 4 and 5.
- c. *Memory Buffer And Location.* The memory necessary for the transceiver consist of two 200-byte output buffers, "tpackbuf" and "pkmess", one 10 x 50 byte input buffer, "pkbuf", and one 50-byte input buffer, "rpakbuf."
- d. *Transfer Protocol.* Serial communication is used for data transmission. To transmit a message, the serial communication status register (SCSR) is polled until the transmitter-ready bit goes high (1). The character to be transmitted is then written to the serial communication data register (SCDR). To receive a message, the serial communication receiver interrupts the processor when a character is received. The interrupt routine implements a finite state machine to recognize valid packets.
- e. *Initiation Condition.* Table 2 indicates the general signal initiation conditions. The HHM-to-PMC RF transceiver interface (IF DRTG) is accessed at 15 second intervals when the HHM sends a CSS request for updated PMC information. Two means exist to initiate the PMC-to-HHM RF transceiver interface (IF DRTH). The interface can be initiated by a data packet sent by the PMC in response to the PMC being activated (PPA), or by a data packet sent by the PMC in response to an information request from the HHM (PSS).
- f. *Priority Level.* This section is not applicable to this interface.
- g. *Expected Response.* The interface table (Table 2) shows the expected responses for all signals.

4. QUALIFICATION REQUIREMENTS

4.1 *General Qualification Requirements.* Both a bench test (hard wired) and a functional test were performed to verify the operation of the PMC-HHM interface. See Appendix I for details of the testing.

4.2 *Special Qualification Requirements.* This section is not applicable to this specification.

6. NOTES

6.1 *Acronyms.* See acronym summary table (Table 6).

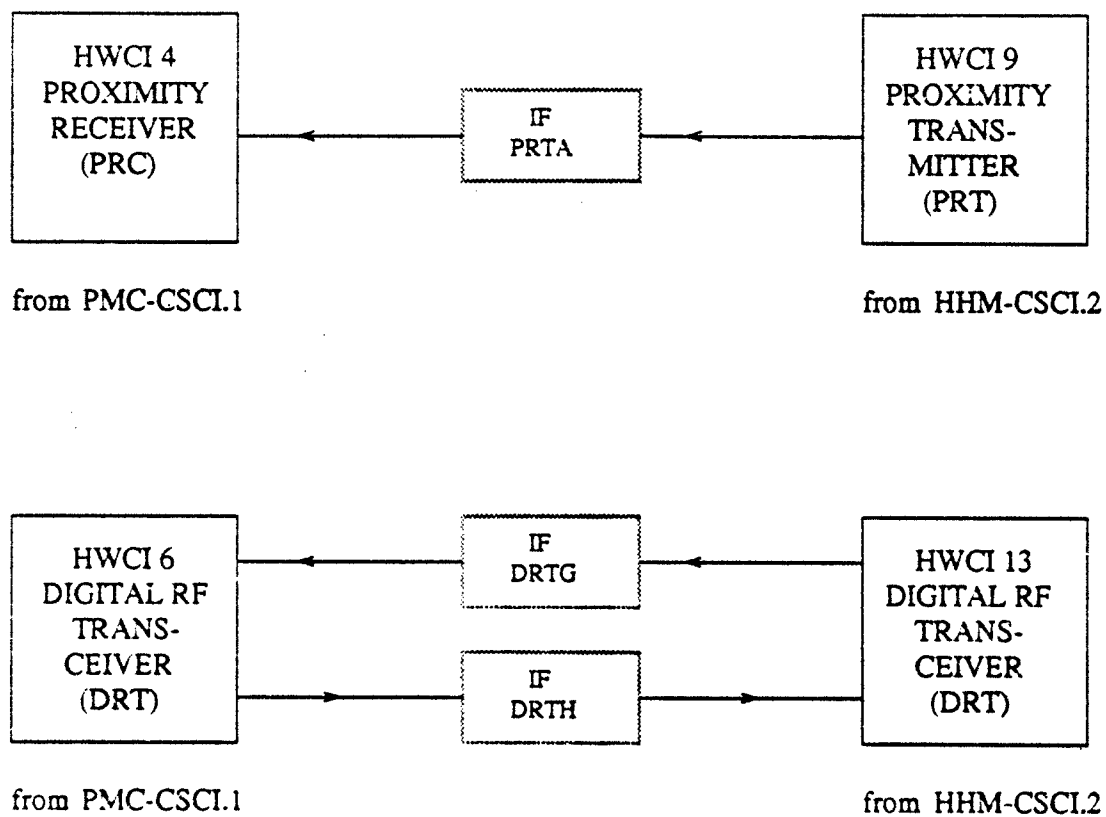


FIGURE 1. Interface block diagram

NOTE:
IF = INTERFACE

TABLE 1. *Interface identification/documentation.*

INTERFACE NAME	INTERFACING ITEM		INTERFACE DOCUMENT NAME	INTERFACE DOCUMENT PARAGRAPH
	TITLE	CI NUMBER		
PRTA	PMC PROXIMITY RECEIVER	4	IRS for PMC-SYS.1 -- PRTA interface	3.3.2.1
	and HHM PROXIMITY TRANSMITTER	9		
DRTG	PMC DIGITAL RF TRANSCEIVER	6	IRS for PMC-SYS.1 -- DRT interfaces	3.3.2.2
	and HHM DIGITAL RF TRANSCEIVER	13		
DRTH	PMC DIGITAL RF TRANSCEIVER	6	IRS for PMC-SYS.1 -- DRT interfaces	3.3.2.2
	and HHM DIGITAL RF TRANSCEIVER	13		

TABLE 2. *Interface summary.*

INTERFACE NAME	INFORMATION DESCRIPTION	INITIATION CONDITION	EXPECTED RESPONSE
PRTA	Proximity data HWCI 9-to-HWCI 4	Proximity receiver activated	PMC sends acknowledge packet
DRTG	Transmit data serial communication HWCI 13-to-HWCI 6	Data packet sent by HHM RF transceiver	PMC processes update request
DRTH	Transmit data serial communication HWCI 6-to-HWCI 13	Data packet sent by PMC RF transceiver	HHM processes packet for display

TABLE 3. *Request for soldier status (CSS) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	CSS	packet type
1 byte*	ASCII		space (blank)
1 byte*	ASCII	I	signals CSCI 1 response
2 bytes	ASCII	hex digits	packet checksum
2 bytes	ASCII	LfCr	packet trailer

* optional

TABLE 4. *PMC positive acknowledge (PPA) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	PPA	packet type
2 bytes	ASCII	hex digits	packet checksum
2 bytes	ASCII	LfCr	packet trailer

TABLE 5. *PMC soldier status (PSS) format.*

DATA LENGTH	DATA TYPE	CONTENT	DESCRIPTION
2 bytes	ASCII	~A	packet header
1 to 5 bytes	ASCII	1 to 5 digits	PMC id number
1 byte	ASCII	:	id end marker
3 bytes	ASCII	PSS	packet type
1 byte	ASCII		space (blank)
1 to 5 bytes	ASCII	1 to 5 digits	evaluation count
1 byte	ASCII		space (blank)
1 to 3 bytes	ASCII	1 to 3 digits	off-wrist level
1 byte	ASCII		space (blank)
2 to 3 bytes	ASCII	2 to 3 digits	heart rate
1 byte	ASCII		space (blank)
1 to 4 bytes	ASCII	1 to 4 digits	motion count
1 byte	ASCII		space (blank)
1 to 3 bytes	ASCII	1 to 3 digits	A/D reference level
1 byte	ASCII		space (blank)
2 bytes	ASCII	LfCr	packet trailer

TABLE 6. *Acronym summary.*

ACRONYM	MEANING
CSS	Request for soldier status
DRT	Digital RF transceiver
DRTG	HHM DRT to PMC DRT interface
DRTH	PMC DRT to HHM DRT interface
HHM	Hand held monitor
PMC	Personal monitor communicator
PPA	PMC positive acknowledge
PRC	PMC proximity receiver
PRTA	PRT to PRC interface
PRT	HHM proximity transmitter
PSS	PMC soldier status

APPENDIX I.

10. INTERFACE TEST SUMMARY

10.1 *Bench Test.*

10.1.1 *Overview of Bench Test.* The serial communication signal lines of a PMC and a HHM were connected directly together. This connection served to bypass the radio transceivers. Temporary pushbutton switches were attached to the PMC proximity interrupt and the HHM activate. Three tests were then performed on the hard wired units.

10.1.2 *General Results of Bench Test.*

10.1.2.1 *Supply Current Test.* A DC millammeter was connected in series with the PMC power supply. The power supply was turned on and the supply current was measured and recorded. The PMC current value was approximately 21 mA. After waiting for one minute, the supply current was again measured and recorded. At this time the PMC CSCI entered the "STOP" state and the PMC current reading was approximately 0.6 mA.

10.1.2.2 *PMC Activation Test.* Actual PMC activation was simulated by pressing the HHM activate pushbutton and the PMC proximity interrupt pushbutton in quick succession. An oscilloscope monitoring the HHM proximity signal output line displayed a burst of 125 KHz square waves when the HHM was activated. The HHM display read

PMC # Activated

where # was the identification number of the PMC. Monitoring the PMC radio power line with an oscilloscope showed the line became active when the PMC was activated.

10.1.2.3 *Status Test.* The HHM would query the activated PMC every 15 seconds. Initially the HHM would receive the message

Off Wrist

The oscilloscope monitoring the radio key line verified proper operation of the line for both the HHM and PMC. The PMC wrist strap was then applied to the wrist of a subject. The subject remained still while the HHM display was monitored. The subject's heart rate was obtained from the subject's pulse. This heart rate was compared to the results displayed by the HHM. The last step was for the

subject to intentionally move during data collection. During this time the HHM display reported motion.

10.2 *Functional Test.*

10.2.1 *Overview of Functional Test.* This test checked the performance and the accuracy of the various PMCs and HHMs. The units were tested by strapping a PMC on a subject and recording the subject's heart rate or motion reading which was displayed by the HHM every 15 seconds. The HHM heart rate was compared with the subject's actual heart rate which was obtained by feeling the subject's pulse. Each PMC was tested continuously for a period of 90 minutes. The testing occurred while the subject was resting, walking, resting after walking, running, and resting after running. All delivered PMCs and HHMs were tested.

10.2.2 *General Results of Functional Test.* The PMCs and HHMs generally performed as expected. Accurate heart rates were obtained when the subject was resting and when the subject was resting after walking. Some heart rates transmitted during light walking were accurate. Most motion caused the HHM to display a motion reading rather than a heart rate value. Running produced the

Heavy Motion

message to appear. Following exercise, the PMC occasionally transmitted heart rates less than 60 bpm where heart rates in the range of 120 bpm were expected. Such readings probably resulted by the impedance signals due to the heavy breathing mixing with the impedance signals due to the heart pumping blood. The HHM had no problem in obtaining information from the PMC

DRAWINGS, ENGINEERING AND ASSOCIATED LISTS
LEVEL 1

FOR THE

PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A007

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

DATA AND DRAWING PACKAGE
FOR ELECTRONIC EQUIPMENT

FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A008

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

SCHEMATIC DIAGRAMS
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A009

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

mm.c

mm.c

```
/* */
/* Multiple Casualty Monitor
/* Ver. 1.0
/* */
/* (c) 1989 Purdue University
/* */

#include <stdio.h>
#include <hlireg.h>
#include <bmech.h>
#include <terminal.h>

/* display handler declarations */
#include "dispdec.h"

#define LF 10
#define CR 13

#define PROXVAL 200
#define PROXMIN 5
#define MAXID 5
#define MAXRETRIES 3
#define EVALTIME 15
/* maximum number of retries */
/* time for PMC to complete an evaluation */

/* Received packet statuses */
#define GOODPAK 1
#define BADCHK 2
#define BADPAK 3
#define TIMEOUT -1

#define ACTIVE 1
#define INACTIVE 0

#define SHOW 1
#define NOSHOW 0

#define MOTIONLT 1
#define MOTIONMOD 2
#define MOTIONHVV 3

/* structure used to set up serial interface */
struct iocbl {ttyb;

char clk50; /* 50 HZ counter for real time clock */
int dcount; /* 50 HZ down counter for general purpose
             * timing */
int stcount; /* 50 HZ down counter for status timing */
int scount; /* 50 HZ down counter for sleep routine */
int etime; /* 1 HZ evaluation time counter */

char tpakbuf[200]; /* buffer to construct packets for
                  * transmission */

char rpakbuf[50]; /* buffer to store received packets */
char *pakptr; /* pointer to receive packet buffer */
int rstate; /* state of receive packet finite state
            * machine */
int chksum; /* check sum calculated on received packets */
int rchksum; /* check sum included in received packets */

char line[80]; /* buffer for display messages */
```

```

/* soldier status values returned from a P.I.C */
int      evnum;          /* evaluation number */
int      offw;           /* PMC off wrist flag */
int      hrate;          /* heart rate */
int      batt;           /* battery voltage */
int      mocount;        /* motion counter */
int      id;             /* PMC ID */

int      adflag;         /* enable A/D conversion flag */
int      adval;          /* value from A/D conversion */

int      retries;        /* number of retries for communication */

int      first;          /* flag to mark first evaluation cycle for a
                          * PMC */

char      '3;

char      clksec;
char      clkmin;
char      clkhr;

/* GG vars */
#define MAXCAS 20
#define WIN_SIZE 10
#define HR_HI 90
#define HR_LOW 50
#define HR_MIN 30

#define TIMEOUT 12
#define DTIMEOUT 8

char      addval;        /* ADD pushbutton current value */
char      delval;        /* DEL pushbutton current value */
char      upval;         /* UP pushbutton current value */
char      dnval;         /* DWN pushbutton current value */
char      hlpval;        /* HLP pushbutton current value */

char      addlast;       /* ADD pushbutton last value */
char      dellast;       /* DEL pushbutton last value */
char      uplast;        /* UP pushbutton last value */
char      dnlast;        /* DWN pushbutton last value */
char      hlpast;        /* HLP pushbutton last value */

char      addstate;      /* ADD pushbutton state */
char      delstate;      /* DEL pushbutton state */
char      upstate;       /* UP pushbutton state */
char      dnstate;       /* DWN pushbutton state */
char      hlpstate;      /* HLP pushbutton state */

char      addflag;       /* ADD pushbutton flag */
char      delflag;       /* DEL pushbutton flag */
char      upflag;        /* UP pushbutton flag */
char      dnflag;        /* DWN pushbutton flag */
char      hlpflag;       /* HLP pushbutton flag */

int      evalflag;       /* evaluate flag */

int      ncas;           /* current number of casualties */
int      curcas;         /* current polled casualty */
int      casptr;         /* casualty pointed to by cursor */

```

mm.c

mm.c

```

int          nextcasid;          /* next usable casid number */
struct cas {
    int      casid;
    int      pmcid;
    int      evnum, offw, hrate, batt, mocount;
    int      inhib;
    struct {
        unsigned hrhi:1, hrlow:1, lowpulse:1, offwrist:1, battlow:1, motion:2;
        unsigned pakerr:1, timeout:1;
    }
}
casualty[MAXCAS];

```

```

char          *motion_str[] = {
    "----",
    "NO ",
    "YES",
    "YES",
    "YES"
};

```

```

char          *stat_str[] = {
    "----",
    "HRT RATE HIGH",
    "HRT RATE LOW",
    "LOW PULSE",
    "OFF WRIST",
    "BATTERY LOW",
    "COM ERROR",
    "COM LOST"
};

```

```

char          null_hr_str[] = "???";

```

```

/* routine declarations */
void          int_setup();
__mod2__ void OOIINT();
__mod2__ void INTINT();

```

```

main()
{

```

main

```

    /* turn rf transmitter off */
    key(0);
    /* initialize keyboard */
    H11PACTL = 0x80 | H11PACTL;
    H11PORTA = 0xD0 | H11PORTA;
    kbclr();
    /* initialize variables */
    casptr = 0;
    ncas = 0;
    nextcasid = 1;
    evalflag = 0;
    curcas = 0;
    /* set up interrupts */
    int_setup();
    /* set up serial port */
    tty_setup();
    /* initialize LCD display */
    disp_init();
    dcursor(0);
    dcursize(3);
    wmake(4, WIN_SIZE);

```

mm.c

mm.c

...main

```

/* Power up message */
help_screen();
pmc_screen();
pmc_show();
/* turn on A/D converter */
adon();
/* main processing loop */
while (1) {
    /* process any button pushes */
    if (addflag) {
        addcas();
        pmc_show();
        addflag = 0;
    } else if (delflag) {
        delcas();
        pmc_show();
        delflag = 0;
    } else if (hlpfld) {
        hlpflag = 0;
        help_screen();
        pmc_screen();
        pmc_show();
    } else if (arrows)
        pmc_show();
    check_cas(SHOW);
    if (adval >= 68 && scount == 0) {
        sprintf(line, "Low Multimonitor Battery");
        dstatus(line);
    }
    stat_clr();
}

sleep(i)
    int i;
{
    scount = i * 50;
    while (scount);
}

prox()
{
    HIIPORTA |= 0x20;
    dcount = 5;
    while (dcount);
    HIIPORTA &= 0xdf;
}

dopss()
{
    /* recieve a PMC Soldier Status */
    int j;
    char s;
    for (j = 0; rpakbuff[j] != '\0'; j++);
    if (strncmp(&rpakb[j + 1], "PSS", 3) != 0)
        return (0);
    j = j + 4;
    s = &rpakbuff[j];
    sscanf(s, "%d %d %d %d %d %d", &evnum, &offw, &hrate, &mocount, &batt);
    casualty|curcas.evnum = evnum;
    casualty|curcas.offw = offw;
    casualty|curcas.hrate = hrate;
    casualty|curcas.mocount = mocount;
    casualty|curcas.batt = batt;
    casualty|curcas.stat.hrhi = 0;
    casualty|curcas.stat.hrlo = 0;
}

```

sleep

prox

dopss

mm.c

mm.c

...dopss

```

casualty[curcas].stat.lowpulse = 0;
casualty[curcas].stat.offwrist = 0;
casualty[curcas].stat.battlow = 0;
casualty[curcas].stat.motion = 1;
if (offw > 190) {
    casualty[curcas].stat.offwrist = 1;
    casualty[curcas].hrate = 0;
    casualty[curcas].mocount = 0;
} else {
    if (mocount >= 1 && mocount < 10) {
        casualty[curcas].stat.motion = 2;
    } else if (mocount >= 10 && mocount < 25) {
        casualty[curcas].stat.motion = 3;
    } else if (mocount >= 25) {
        casualty[curcas].stat.motion = 4;
    }
    if ((hrate == 0) || (hrate == -4)) {
        casualty[curcas].stat.lowpulse = 1;
    } else if ((hrate > 0) && (hrate <= HR_LOW)) {
        casualty[curcas].stat.hriscw = 1;
        alarm(2);
    } else if (hrate >= HR_HI) {
        casualty[curcas].stat.hrhi = 1;
        alarm(3);
    }
}
if (batt >= 68) {
    casualty[curcas].stat.battlow = 1;
}
return (1);
}

doppa()
{
    /* do Positive Packet Acknowledge */
    int j;
    char *s;
    int lid;
    for (j = 0; rpakbuf[j] != '\0'; j++);
    if (strncmp(&rpakbuf[j] + 1, "PPA", 3) != 0)
        return (0);
    j--;
    while (isdigit(rpakbuf[j] - 1))
        j--;
    s = &rpakbuf[j];
    sscanf(s, "%d", &lid);
    return (lid);
}

docss(id)
{
    int id;
    /* send out Central Soldier Status query */
    char *p;
    p = &tpakbuf[0];
    sprintf(p, "A%d:CSS I", id);
    sendpak(p);
}

sendpak(pk)
{
    char *pk;
    {
        int i;
        char *p;
        p = pk;
        for (i = 0; *p; p++)
            i = i + *p;
    }
}

```

doppa

docss

sendpak

mm.c

```

i = i & 0xff;
*p = (i >> 4) + '0';
if (*p > '9')
    *p = *p + 7;
p++;
*p = (i & 0xf) + '0';
if (*p > '9')
    *p = *p + 7;
p++;
*p = LF;
p++;
*p = CR;
p++;
*p = 0;
key(1);
dcount = 10;
/* while(dcount); */
waitd();
p = pk;
pkputc('\r');
pkputc('X');
while (*p) {
    pkputc(*p);
    p++;
}
pkputc(CR);
pkputc(CR);
key(0);
}

pkputc(c)
    char    c;
{
    putchar(c);
}

getpacket(timeout, pktype, id)
    int      timeout;
    char     *pktype;
    int      id;
{
    int      done, c, j, lid;
    char     *s;
    rstate = 0;
    chksum = 0;
    pakptr = &rpakbuf[0];
    count = timeout * 50;
    done = 0;
    while (1) {
        while (com_rdy() == 0)
            if (checkd() == 0)
                return (TIMEOUT);
        c = getchar() & 0x7f;
        /* dputc(c); */
        /* dctrl(0x10); */
        switch (rstate) {
            case 0:
                if (c == '\n') {
                    *pakptr++ = c;
                    chksum += c;
                    rstate = 1;
                }
                break;
            case 1:
                if (c == '\n')

```

mm.c

...sendpak

pkputc

getpacket

mm.c

mm.c

...getpacket

```

        break;
    if (c == 'A') {
        *pakptr++ = c;
        chksum += c;
        rstate = 2;
    } else
        pkabort();
    break;
case 2:
    if (c == LF || c == CR) {
        *pakptr = 0;
        s = pakptr;
        s--;
        s--;
        scanf(s, "%x", &rchksum);
        chksum -= *s++;
        chksum -= *s;
        chksum = chksum & 0xff;
        *s-- = 0;
        *s = 0;
        if (chksum != rchksum)
            pkabort();
        else {
            for (j = 0; rpakbuf[j] != '\0'; j++);
            if (strcmp(&rpakbuf[j + 1], pkttype, 3) == 0) {
                j--;
                while (isdigit(rpakbuf[j - 1]))
                    j--;
                s = &rpakbuf[j];
                scanf(s, "%d", &lid);
                if ((lid == id) || (id == 0))
                    return (GOODPAK);
            }
        }
        /* not our packet, toss it away */
        pkabort();
    } else {
        *pakptr++ = c;
        chksum += c;
    }
    break;
}
if (pakptr >= &rpakbuf[49])
    pkabort();
}
)

```

```

pkabort()
{
    pakptr = &rpakbuf[0];
    chksum = 0;
    rstate = 0;
}

```

```

mod2 void
INTINT()
{
}

```

```

key(i)
int
{
    if (i == 0)
        H11PORTA |= 0x08;
    else

```

pkabort

INTINT

key

mm.c

```
    H11PORTA &= 0xF7;
}

mod2 void
OC1INT()
{
    H11TFLG1 = 0x80;
    H11TOC1 += 0x9c40;
    H11COPRST = 0x55;
    H11COPRST = 0xaa;
    /* check and debounce keys (keys are active low) */
    if (((H11PORTA & 0x04) == 0))
        blpval = ACTIVE;
    else
        blpval = INACTIVE;
    if (clk50 & 0x01) {
        if (((H11PORTA & 0x01) == 0))
            dnval = ACTIVE;
        else
            dnval = INACTIVE;
        if (((H11PORTA & 0x02) == 0))
            upval = ACTIVE;
        else
            upval = INACTIVE;
        /* set up for next read */
        H11PORTA = 0x40 | (H11PORTA & 0x3F);
    } else {
        if (((H11PORTA & 0x01) == 0))
            delval = ACTIVE;
        else
            delval = INACTIVE;
        if (((H11PORTA & 0x02) == 0))
            addval = ACTIVE;
        else
            addval = INACTIVE;
        /* set up for next read */
        H11PORTA = 0x80 | (H11PORTA & 0x3F);
    }
    if (blpval + dnval + upval + delval + addval > 1) {
        blpval = INACTIVE;
        dnval = INACTIVE;
        upval = INACTIVE;
        delval = INACTIVE;
        addval = INACTIVE;
    }
    dokey(&blpval, &blplast, &blpstate, &blpflag);
    dokey(&dnval, &dnlast, &dnstate, &dnflag);
    dokey(&upval, &uplast, &upstate, &upflag);
    dokey(&delval, &deplast, &destate, &deflag);
    dokey(&addval, &addlast, &addstate, &addflag);
    if (adflag && clk50 == 0) {
        H11ADCTL = 0x10;
        while (H11ADCTL & 0x80 == 0);
        adval = H11ADR1;
    }
    if (dcount)
        dcount--;
    if (stcount)
        stcount--;
    if (scount)
        scount--;
    clk50++;
    if (clk50 >= 50) {
        clksec++;
    }
}
```

mm.c

...key

OC1INT

mm.c

```
    clk50 = 0;
    etime++;
}
if (clksec >= 60) {
    clkmin++;
    clksec = 0;
}
if (clkmin >= 60) {
    clkhr++;
    clkmin = 0;
}
}
```

dokey(val, last, state, flag)

```
    char      *val, *last, *state, *flag;
{
    if (*state == INACTIVE) {
        if (*val == ACTIVE && *last == ACTIVE) {
            *state = ACTIVE;
            *flag = ACTIVE;
        }
    } else {
        if (*val == INACTIVE && *last == INACTIVE && *flag == INACTIVE) {
            *state = INACTIVE;
        }
    }
    *last = *val;
}
```

mod2 void
COPINT()

```
{
    dstatus0("COP RESET!");
}
```

void

int_setup()

```
{
    __mod2__ void OC1INT();
    __mod2__ void INTINT();
    __mod2__ void COPINT();
    void VECTOR();
    dcount = 0;
    scount = 0;
    stcount = 0;
    etime = 0;
    VECTOR(OC1INT, 9);
    VECTOR(INTINT, 14);
    VECTOR(COPINT, 17);
    VECTOR(COPINT, 18);
    HIITOC1 = 0;
    HIITMSK1 |= 0x80;
}
```

tty_setup()

```
{
    chksum = 0;
    rchksum = 0;
    rstate = 0;
    open();
    HI1BAUD = 0x35;
    ioctl(ttyb, GETPARAMS);
    ttyb->io_erase = CTRL('H');
    ttyb->io_kill = CTRL('U');
```

mm.c

...OC1INT

dokey

COPINT

int_setup

tty_setup

mm.c

```

    ttyb->io_flags = RAW;
    ioctl(ttyb, SETPARAMS);
}

com_rdy()
{
    if (H11SCSR & 0x20)
        return (1);
    else
        return (0);
}

adon()
{
    H11OPTION |= 0x80;          /* power up a/d */
    H11PORTA |= 0x10;          /* power up wristz circuit */
    adflag = 1;
}

adoff()
{
    H11OPTION &= 0x7F;          /* power down a/d */
    H11PORTA &= 0xEF;          /* power up wristz circuit */
    adflag = 0;
}

/*
 *
 * GG routines
 *
 */

help_screen()
{
    ncls();
    nsetattr(DISP_NORM);
    nmessage(0, 0, "Multiple Casualty Monitor");
    nmessage(1, 0, " ");
    nmessage(2, 0, "INSTRUCTIONS");
    nmessage(3, 0, " ");
    nsetattr(DISP_REV);
    nmessage(3, 0, "To Add Casualty To List.");
    nsetattr(DISP_NORM);
    nmessage(4, 0, "Put MM near PMC, Press <ADD>");
    nmessage(5, 0, "Wait for \"PMC ACTIVATED\"");
    nmessage(6, 0, "Label body with number shown");
    nmessage(7, 0, " ");
    nsetattr(DISP_REV);
    nmessage(7, 0, "To Delete Casualty from List");
    nsetattr(DISP_NORM);
    nmessage(8, 0, "Use arrow keys to select");
    nmessage(9, 0, "casualty to be deleted");
    nmessage(10, 0, "Press <DELETE>, the message");
    nmessage(11, 0, "\"CONFIRM DELETION\" appears");
    nmessage(12, 0, "Press <DELETE> again");
    nmessage(13, 0, " ");
    nsetattr(DISP_REV);
    nmessage(13, 0, "To See This Screen.");
    nsetattr(DISP_NORM);
    nmessage(14, 0, "Press <HELP>");
    nsetattr(DISP_RBLINK);
    nmessage(15, 0, "Press any key to exit HELP");
    nsetattr(DISP_NORM);
    dupdate();
    while ('keyhit()')

```

mm.c

...tty_setup

com_rdy

adon

adoff

help_screen

mm.c

mm.c

...help_screen

```

        check_cas(NOSHOW);
        sleep(1);
        if (addflag) {
            kbcir();
            addflag = TRUE;
        } else
            kbcir();
        stcount = TIMEOUT * 50;
    }

pmc_screen()
{
    nsetattr(DISP_REV);
    nmessage(0, 0, "Multiple Casualty Monitor  ");
    nsetattr(DISP_NORM);
    nmessage(1, 0, "CAUS HEART MOT.  STATUS  ");
    nmessage(2, 0, "NO. RATE  ");
    nmessage(3, 0, "-----");
    nmessage(14, 0, "-----");
    nsetattr(DISP_REV);
    nmessage(15, 0, "Press <HELP> for instructions ");
    nsetattr(DISP_NORM);
    wsetattr(DISP_NORM);
    wcls();
}

```

pmc_screen

```

pmc_show()
{
    int i;
    int n;
    int status;
    char hr_str[5];
    char *hr_sptr;

    /* clear window if necessary */
    if (n == offwin(casptr)) {
        if (n > 0)
            wscrollup(n);
        else
            wscrolldown(-n);
    }
    /* display casualties */
    domore();
    for (i = 0; i < ncas; i++) {
        wsettextpos(i, 0);
        if (status == maxstat(i)) {
            if (i == casptr)
                wsetattr(DISP_RBLINK);
            else
                wsetattr(DISP_BLINK);
        } else {
            if (i == casptr)
                wsetattr(DISP_REV);
            else
                wsetattr(DISP_NORM);
        }
        if (casualty[i].brate > HR_MIN) {
            sprintf(hr_str, "%3d", casualty[i].brate);
            hr_sptr = hr_str;
        } else {
            hr_sptr = null_hr_str;
        }
        sprintf(line, " %03d %3s %3s %14s", casualty[i].casid,
            hr_sptr, motion_str[casualty[i].stat.motion],
            stat_str[status]);
    }
}

```

pmc_show

mm.c

```

        wputs(line);
    }
    wupdate();
}

pmc_show1(i)
    int i,
    int status;
    char hr_str[5];
    char *hr_sptr,

/* alter window if neccessary */
/*
 * if(n = offwin(casptr)){ if(n > 0) wscrollup(n); else wscrolldown(-n);
 */
/* update casualty window */
wsettextpos(i, 0);
if (status == maxstat(i)) {
    if (i == casptr)
        wsetattr(DISP_RBLINK);
    else
        wsetattr(DISP_BLINK);
} else {
    if (i == casptr)
        wsetattr(DISP_REV);
    else
        wsetattr(DISP_NORM);
}
if (casualty[i] hrate > 30) {
    sprintf(hr_str, "%3d", casualty[i] hrate);
    hr_sptr = hr_str;
} else {
    hr_sptr == null_hr_str;
}
sprintf(line, " %03d %3s %3s %-14s", casualty[i] casid,
    hr_sptr, motion_str[casualty[i] stat.motion],
    stat_str[status]);
wputs(line);
wupdlw(i);
}

```

```

dstatus(s)
    char *s,

/* display status message (full line) */
{
    nsetattr(DISP_REV);
    nsettextpos(15, 0);
    nputs(" ");
    nsettextpos(15, 0);
    nputs(s);
    dupdln(15);
    stcount = STIMEOUT * 50;
}

```

```

dstatus0(s)
    char *s,

/* display primary status message (half line) */
{
    nsetattr(DISP_REV);
    nsettextpos(15, 0);
    nputs(" ");
    nsettextpos(15, 0);
    nputs(s);
    dupdln(15);
    stcount = STIMEOUT * 50;
}

```

mm.c

...pmc_show

pmc_show1

dstatus

dstatus0

mm.c

mm.c

...dstatus0

dstatus1

stat_clr

offwin

inwin

domore

```

}

dstatus1(s)
{
    char s;

    /* display secondary status message (half
     * line) */
    nsetattr(DISPLAY_REV);
    nsettextpos(15, 15);
    nputs(" ");
    nsettextpos(15, 15);
    nputs(s);
    dupln(15);
    stcount = STIMEOUT * 50;
}

stat_clr()
{
    if (stcount == 0) {
        nsetattr(DISPLAY_REV);
        nsettextpos(15, 0);
        nputs(" ");
        dupln(15);
    }
}

offwin(pos)
{
    int pos;

    /* return scroll to get window pos on screen */
    int i;

    i = wgetorg();
    if (pos < i)
        return (pos - i); /* sel dn neg */
    if (pos > WIN_SIZE + i - 1)
        return (pos - (WIN_SIZE + i - 1));
    return (0);
}

inwin(pos)
{
    int pos;

    /* return display row of window pos (-1 if
     * not vis) */
    int i;

    i = wgetorg();
    if (pos < i)
        return (-1);
    if (pos > WIN_SIZE + i - 1)
        return (-1);
    return (Win_row_dorg + pos - i);
}

domore()
{
    int i;

    i = wgetorg();
    nsetattr(DISPLAY_NORM);
    if (i > 0) {
        nmessage(3, 0, "-----More-----");
        nputattr(3, 12, DISPLAY_REV);
        nputattr(3, 13, DISPLAY_REV);
        nputattr(3, 14, DISPLAY_REV);
        nputattr(3, 15, DISPLAY_REV);
    } else

```

mm.c

mm.c

...domiore

```
        nmessage(3, 0, "-----");
    if (ncas > i + WIN_SIZE) {
        nmessage(14, 0, "v-----v-More-v-----v");
        nputattr(14, 12, DISP_REV);
        nputattr(14, 13, DISP_REV);
        nputattr(14, 14, DISP_REV);
        nputattr(14, 15, DISP_REV);
    } else
        nmessage(14, 0, "-----");
}
```

/* key board routines */

keyhit()

```
{
    if (hlplflag || dnflflag || upflflag || delflflag || addflflag)
        return (1);
    else
        return (0);
}
```

kbclr()

```
{
    addlast = INACTIVE;
    dellast = INACTIVE;
    uplast = INACTIVE;
    dnlast = INACTIVE;
    hlplast = INACTIVE;
    addflflag = INACTIVE;
    delflflag = INACTIVE;
    upflflag = INACTIVE;
    dnflflag = INACTIVE;
    hlplflag = INACTIVE;
    addstate = INACTIVE;
    delstate = INACTIVE;
    upstate = INACTIVE;
    dnstate = INACTIVE;
    hlplstate = INACTIVE;
}
```

arrows()

```
{
    if (upflflag || dnflflag) {
        if (upflflag) {
            upflflag = 0;
            if (casptr > 0)
                casptr--;
        } else if (dnflflag) {
            dnflflag = 0;
            if (casptr < ncas - 1)
                casptr++;
        }
        return (1);
    }
    return (0);
}
```

addcas()

```
{
    int          status;
    int          n;

    dstatus("Attempting PMC Activation");
    if (ncas >= MAXCAS) {
        dstatus("Limit Exceeded - PMC Not Added");
    }
}
```

keyhit

kbclr

arrows

addcas

mm.c

mm.c

...addcas

```

        sleep(1);
        return;
    }
    /* attempt PMC activation */
    retries = 0;
    do {
        /* pulse proximity transmitter */
        prox();
        /* check for PMC response */
        status = getpacket(2, "PPA", 0);
#ifdef BENCH
        dstatus(rpakbuf);
#endif
        if (status == GOODPAK)
            id = doppa();
        retries++;
        /* repeat activation attempt until successful or too */
        /* many retries */
    } while ((status != GOODPAK || id == 0) && retries < MAXRETRIES);
    if (retries >= MAXRETRIES)
        id = 0;
    addflag = 0;
    /* if id == 0 then activation was unsuccessful */
    if (id == 0) {
        dstatus("ERROR--Repeat Activate Steps");
        sleep(1);
        dstatus("Put MM Near PMC, Press <ADD>");
    } else if ((n = inlist(id)) != -1) {
        /* previously activated PMC */
        if (casualty[n].stat.timeout == 1) {
            dstatus("Casualty Reactivated");
            casualty[n].stat.timeout = 0;
            sleep(1);
        } else {
            dstatus("Casualty Already Active");
            sleep(1);
        }
    }
    } else {
        casualty[ncas].pmcid = id;
        casualty[ncas].casid = nextcasid;
        casualty[ncas].evnum = 0;
        casualty[ncas].offw = 0;
        casualty[ncas].brate = 0;
        casualty[ncas].batt = 0;
        casualty[ncas].mocount = 0;
        casualty[ncas].inhib = 1;
        casualty[ncas].stat.hrhi = 0;
        casualty[ncas].stat.brlo = 0;
        casualty[ncas].stat.lowpulse = 0;
        casualty[ncas].stat.offwrist = 0;
        casualty[ncas].stat.battlow = 0;
        casualty[ncas].stat.motion = 0;
        casualty[ncas].stat.pakerr = 0;
        casualty[ncas].stat.timeout = 0;
        casptr = ncas++;
        sprintf(line, "PMC %d ACTIVATED", id);
        dstatus(line);
        sleep(1);
        sprintf(line, "MARK CASUALTY WITH #%-03d", nextcasid++);
        dstatus(line);
    }
}

delcas()
{

```

delcas

mm.c

mm.c

...delcas

```

if (ncas > 0) {
    /* confirm delete */
    dstatus("Hit <DELETE> to Confirm");
    delflag = 0;
    dcount = DTIMEOUT * 50;
    while ('keyhit()' && dcount);
    if (delflag) {
        removecas();
        dstatus("");
    } else
        dstatus("Casualty Not Deleted");
} else
    dstatus("No Casualties to Delete");
}

removecas()
{
    int          i, j;

    ncas--;
    for (i = casptr; i < ncas; i++) {
        j = i + 1;
        casualty[i].casid = casualty[j].casid;
        casualty[i].pmcid = casualty[j].pmcid;
        casualty[i].evnum = casualty[j].evnum;
        casualty[i].offw = casualty[j].offw;
        casualty[i].hrate = casualty[j].hrate;
        casualty[i].batt = casualty[j].batt;
        casualty[i].mcount = casualty[j].mcount;
        casualty[i].stat.hrhi = casualty[j].stat.hrhi;
        casualty[i].stat.hrlo = casualty[j].stat.hrlo;
        casualty[i].stat.lowpulse = casualty[j].stat.lowpulse;
        casualty[i].stat.offwrist = casualty[j].stat.offwrist;
        casualty[i].stat.battlow = casualty[j].stat.battlow;
        casualty[i].stat.motion = casualty[j].stat.motion;
        casualty[i].stat.pakerr = casualty[j].stat.pakerr;
        casualty[i].stat.timeout = casualty[j].stat.timeout;
    }
    wsetattr(DISP_NORM);
    wcin(ncas);
    if (casptr > ncas - 1)
        casptr = ncas - 1;
}

```

removecas

```

check_cas(showmode)
{
    int          showmode;

    int          cursrow;

    if (etime >= EVALTIME) {
        evalflag = 1;
        etime = 0;
    }
    if (evalflag) {
        if (curcas < ncas) {
            if (casualty[curcas].inhib)
                casualty[curcas].inhib = 0;
            else
                getstatus(casualty[curcas].pmcid);
            if (showmode && ((cursrow = inwin(curcas)) != -1)) {
                /* do cursor, get and show status */
                dsetcurspos(cursrow, 0);
                dcursor(1);
                pmc_showl(curcas);
            }
        }
    }
}

```

check_cas

mm.c

mm.c

...check_cas

```
        dcursor(0);
    }
    curcas++;
} else {
    curcas = 0;
    evalflag = 0;
}
}
```

getstatus

```
getstatus(id)
{
    int          id;

    int          status;

    retries = 0;
    casualty[curcas].stat.pakerr = 0;
    casualty[curcas].stat.timeout = 0;
    do {
        docss(id);
        status = getpacket(2, "PSS", id);
        retries++;
    } while (status != GOODPAK && retries < MAXRETRIES);
    if (retries < MAXRETRIES)
        dopss();
    else {
        casualty[curcas].stat.timeout = 1;
    }
}
```

inlist

```
inlist(id)
{
    int          id;

    int          i;

    /* return casid if pmcid is in casualty list */

    for (i = 0; i < ncas; i++) {
        if (casualty[i].pmcid == id)
            return (i);
    }
    return (-1);
}
```

alarm

```
alarm(nbeeps)
{
    int          nbeeps;

    while (nbeeps-- > 0) {
        dcount = 20;
        H11PORTA = H11PORTA & 0xef;
        waitd();
        H11PORTA = H11PORTA | 0x10;
        if (nbeeps > 0) {
            dcount = 20;
            waitd();
        }
    }
}
```

maxstat

```
maxstat(i)
{
    int          i;

    if (casualty[i].stat.timeout == 1)
        return (7);
    if (casualty[i].stat.pakerr == 1)
        return (6);
    if (casualty[i].stat.offwrist == 1)
```

mm.c

```
        return (4);
    if (casualty[i].stat.hrlow == 1)
        return (2);
    if (casualty[i].stat.hrhi == 1)
        return (1);
    if (casualty[i].stat.lowpulse == 1)
        return (3);
    if (casualty[i].stat.battlow == 1)
        return (5);
    return (0);
}
```

waitd()

```
{
    while (acount)
        if (arrows())
            pmc_show();
}
```

checkd()

```
{
    if (dcount) {
        if (arrows())
            pmc_show();
        return (1);
    }
    return (0);
}
```

/ check dcount to see if still active */*

mm.c

...mazstat

waitd

checkd

displib.c

```

/*****
/* dcln.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dcln(line)
    int        line;
{
    int        i;

    dsettextpos(line, 0);
    for (i = 0; i < DISP_COLS; i++)
        dputc(' ');
    dsettextpos(0, 0);
}
/*****/

```

```

/* dcls.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dcls()
{
    int        i, j;

    for (i = 0; i < DISP_ROWS; i++) {
        dsettextpos(i, 0);
        for (j = 0; j < DISP_COLS; j++)
            dputc(' ');
    }
    dsettextpos(0, 0);
}
/*****/

```

```

/* dctrl.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dctrl(v)
    int        v;
{
    /* send control byte to display */

    DISPCWAIT;
}

```

displib.c

*/

dcln

*/

dcls

*/

dctrl

displib.c

displib.c

...dctrl

```
DISPCTL = v
}
/*****
/* dctrl1.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/
```

*/

```
#include <display.h>
```

dctrl1

```
dctrl1(v, data)
    int v, data;
{
    /* send display one data byte */
    DISPWAIT;
    /* and one control byte
    DISPDATA = data;
    DISPWAIT;
    DISPCTL = v;
```

```
}
/*****
/* dctrl2.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/
```

*/

```
#include <display.h>
```

dctrl2

```
dctrl2(v, data1, data2)
    int v, data1, data2;
{
    /* send display two data bytes */
    DISPWAIT;
    /* and one control byte
    DISPDATA = data1;
    DISPWAIT;
    DISPDATA = data2;
    DISPWAIT;
    DISPCTL = v;
```

```
}
/*****
/* dcursor.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/
```

*/

```
#include <display.h>
```

dcursor

```
dcursor(mode)
    int mode;
{
```


displib.c

```
    if (mode)
        dctrl(0x9E);          /* DISPLAY CURSOR ON */

    else
        dctrl(0x9C);          /* DISPLAY CURSOR OFF */
}

/*****
/* dcursor.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dcursor(mode)
    int      mode;

{

    if (mode)
        dctrl(0x9E);          /* DISPLAY CURSOR ON */

    else
        dctrl(0x9C);          /* DISPLAY CURSOR OFF */
}

/*****
/* dcurssize.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dcurssize(size)
    int      size;

{

    size = 0xA0 | (size & 0x07);

    dctrl(size);              /* CURSOR TYPE */
}

/*****
/* ddata.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
ddata(c)
    int      c;

```

displib.c

...dcursor

dcursor

dcurssize

*/

*/

ddata

displib.c

```
{
    /* send data byte to display */
    DISPWAIT;
    DISPDATA = c;
}
/*.....*/
/* disp_init.c:
/* AND1091 LCD display routine for use with the Multimonitor:
/*.....*/
#include <display.h>
disp_init()
{
```

```
    int            i, j;

    while ((DISPCTL & 0x20) == 0);
    dctrl(0x84);          /* MODE SET */
    dctrl2(0x40, 0, 0x0A); /* TXT HOME COM */
    dctrl2(0x41, 0x1e, 0); /* TXT AREA COM */
    dctrl2(0x42, 0, 0x08); /* GR HOME COM */
    dctrl2(0x43, 0x1e, 0); /* GR AREA COM */
    dctrl2(0x24, 0, 0x0A); /* ADDR PTR SET */
    dctrl2(0x24, 0, 0x08); /* ADDR PTR SET */
    dctrl(0xA0);          /* CURSOR TYPE */
    dctrl(0x9C);          /* DISPLAY MODE SET */

    Disp_text_row = 0;
    Disp_text_col = 0;
    Disp_curs_row = 0;
    Disp_curs_col = 0;
    Disp_text_attr = DISP_NORM;
    Nxt_text_row = 0;
    Nxt_text_col = 0;
    Nxt_curs_row = 0;
    Nxt_curs_col = 0;
    Nxt_text_attr = DISP_NORM;
    for (i = 0; i < DISP_ROWS; i++) {
        for (j = 0; j < DISP_COLS; j++) {
```

displib.c

...ddata

disp_init

displib.c

```
Cur_img_ch[i][j] = 0xFF;
Cur_img_at[i][j] = 0xFF;
    }
    ncls();
    dupdate();
}

/*****
/* dmmessage.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dmmessage(r, c, s)
    int      r, c;
    char     *s;
{
    dsettextpos(r, c);
    dputs(s);
}

/*****
/* dputattr.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dputattr(r, c, attr)
    int      r, c, attr;
{
    dsetattrpos(r, c);
    dctrl1(0XC0, attr);
}

/*****
/* dputc.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dputc(c)
    int      c;
{
```

displib.c

...disp_init

dmmessage

dputattr

dputc

displib.c

```

    if (c == '\n') {
        Disp_text_col = 0;
        Disp_text_row = (Disp_text_row + 1) % DISP_ROWS;
    }
    else {
        c = c - 0x20;

        /* write char */
        dsettextpos(Disp_text_row, Disp_text_col);

        dctrl(0xc0, c);

        /* write char's attribute */
        dsetattrpos(Disp_text_row, Disp_text_col);

        dctrl(0xc0, Disp_text_attr);

        /* update row and col variables */
        Disp_text_col++;

        if (Disp_text_col >= DISP_COLS) {
            Disp_text_col = 0;
            Disp_text_row++;

            if (Disp_text_row >= DISP_ROWS) {
                Disp_text_row = 0;
            }
        }
    }
}

/* ..... */
/* dputchar c: */
/* AND1091 LCD display routine for use with the Multimonitor */
/* ..... */

#include <display.h>
dputchar(r, c, ch)
    int    r, c, ch;
{
    ch = ch - 0x20;

    dsettextpos(r, c);

    dctrl(0xc0, ch);
}

/* ..... */
/* dputs c: */

```

displib.c

...dputc

dputchar

displib.c

```
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****
```

```
#include <display.h>
```

```
dputs(s)
char *s;
{
```

```
while (*s) {
```

```
    dputc(*s);
```

```
    s++;
```

```
}
```

```
} /*****/
```

```
/* drc2aaddr.c:
```

```
/* AND1091 LCD display routine for use with the Multimonitor: */
```

```
*****/
```

```
#include <display.h>
```

```
drc2aaddr(r, c, msb_ptr, lsb_ptr)
int r, c, *msb_ptr, *lsb_ptr;
{
```

```
if (r < 8) {
```

```
    *lsb_ptr = r * 30 + c;
```

```
    *msb_ptr = DISP_AADDR1M;
```

```
}
```

```
else {
```

```
    *lsb_ptr = (r - 8) * 30 + c;
```

```
    *msb_ptr = DISP_AADDR2M;
```

```
}
```

```
} /*****/
```

```
/* drc2taddr.c
```

```
/* AND1091 LCD display routine for use with the Multimonitor: */
```

```
*****/
```

```
#include <display.h>
```

```
drc2taddr(r, c, msb_ptr, lsb_ptr)
int r, c, *msb_ptr, *lsb_ptr;
{
```

```
if (r < 8) {
```

```
    *lsb_ptr = r * 30 + c;
```

```
    *msb_ptr = DISP_TADDR1M;
```

displib.c

dputs

drc2aaddr

drc2taddr

displib.c

```

    }
    else {
        *lsb_ptr = (r - 8) * 30 + c;
        *msb_ptr = DISP_TADDR2M;
    }
}
/*****
/* dsetattrpos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dsetattrpos(r, c)
    int r, c;
{
    int lsb, msb;

    drc2aaddr(r, c, &msb, &lsb);
    dctrl2(0X24, lsb, msb);

    Disp_text_row = r;
    Disp_text_col = c;
}
/*****
/* dsetattr.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dsetattr(attr)
    int attr;
{
    Disp_text_attr = attr;
}
/*****
/* dsetcurspos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dsetcurspos(r, c)
    int r, c;
{

```

displib.c

...rc2taddr

dsetattrpos

dsetattr

dsetcurspos

displib.c

```
if (r >= 8)
r = r + 0X10 - 0X08;

dctrl2(0X21, c, r);

Disp_curs_row = r;
Disp_curs_col = c;
}
/* ..... */
/* dsettextpos.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */
```

```
#include <display.h>
```

```
dsettextpos(r, c)
int r, c;
{
    int isb, msb;

    drc2addr(r, c, &msb, &isb);
    dctrl2(0X24, isb, msb);
    Disp_text_row = r;
    Disp_text_col = c;
}
```

```
/* ..... */
/* dupdate.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */
```

```
#include <display.h>
```

```
dupdate()
{
    int i, j;

    for (i = 0; i < DISP_ROWS; i++) {
        for (j = 0; j < DISP_COLS; j++) {
            if (Cur_img_ch[i][j] != Nxt_img_ch[i][j]) {
                dsettextpos(i, j);
                dputchar(i, j, Nxt_img_ch[i][j]);
                Cur_img_ch[i][j] = Nxt_img_ch[i][j];
            }
            if (Cur_img_at[i][j] != Nxt_img_at[i][j]) {
```

displib.c

...dsetcurspos

dsettextpos

dupdate

displib.c

```

        dsetattrpos(i, j);
        dputattr(i, j, Nxt_img_at[i][j]);
        Cur_img_at[i][j] = Nxt_img_at[i][j];
    }
}
Disp_text_row = Nxt_text_row;
Disp_text_col = Nxt_text_col;
Disp_curs_row = Nxt_curs_row;
Disp_curs_col = Nxt_curs_col;
Disp_text_attr = Nxt_text_attr;
dsetcurspos(Nxt_curs_row, Nxt_curs_col);
}
/*****
/* dupdln.c
/* AND1091 LCD display routine for use with the Multimonitor: */
*****/

#include <display.h>
dupdln(i)
    int i;
{
    int j;

    for (j = 0; j < DISP_COLS; j++) {
        if (Cur_img_ch[i][j] != Nxt_img_ch[i][j]) {
            dsettextpos(i, j);
            dputchar(i, j, Nxt_img_ch[i][j]);
            Cur_img_ch[i][j] = Nxt_img_ch[i][j];
        }
        if (Cur_img_at[i][j] != Nxt_img_at[i][j]) {
            dsetattrpos(i, j);
            dputattr(i, j, Nxt_img_at[i][j]);
            Cur_img_at[i][j] = Nxt_img_at[i][j];
        }
    }
    Disp_text_row = Nxt_text_row;
    Disp_text_col = Nxt_text_col;
    Disp_curs_row = Nxt_curs_row;

```

displib.c

...dupdate

dupdln

displib.c

displib.c

...dupdln

```

Disp_curs_col = Nxt_curs_col;
Disp_text_attr = Nxt_text_attr;
dsetcurspos(Nxt_curs_row, Nxt_curs_col);
}
/* ncln.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
#include <display.h>
ncln(line)
int line;
{
    int i;

    nsettextpos(line, 0);
    for (i = 0; i < DISP_COLS; i++)
        nputc(' ');
    nsettextpos(0, 0);
}
/* ncls.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
#include <display.h>
ncls()
{
    int i, j;

    for (i = 0; i < DISP_ROWS; i++) {
        nsettextpos(i, 0);
        for (j = 0; j < DISP_COLS; j++)
            nputc(' ');
    }
    nsettextpos(0, 0);
}
/* nmessage.c: */

```

ncln

ncls

displib.c

```
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****/

#include <display.h>
nmessage(r, c, s)
    int      r, c;
    char      *s;
{
    nsettextpos(r, c);
    nputs(s);
}
/*****/

/* nputattr.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****/

#include <display.h>
nputattr(r, c, attr)
    int      r, c, attr;
{
    Nxt_img_at[r][c] = attr;
}
/*****/

/* nputc.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****/

#include <display.h>
nputc(c)
    int      c;
{
    if (c == '\n') {
        Nxt_text_col = 0;
        Nxt_text_row = (Nxt_text_row + 1) % DISP_ROWS;
    }
    else {
        /* write char */
        nsettextpos(Nxt_text_row, Nxt_text_col);

        /* write char's attribute */
        nsetattrpos(Nxt_text_row, Nxt_text_col);

        Nxt_img_ch[Nxt_text_row][Nxt_text_col] = c;
        Nxt_img_at[Nxt_text_row][Nxt_text_col] = Nxt_text_attr;
    }
}
```

displib.c

nmessage

nputattr

nputc

displib.c

```

/* update row and col variables */
Nxt_text_col++;

if (Nxt_text_col >= DISP_COLS) {
    Nxt_text_col = 0;
    Nxt_text_row++;
    if (Nxt_text_row >= DISP_ROWS) {
        Nxt_text_row = 0;
    }
}

}

/* nputchar.c
/* AND1091 LCD display routine for use with the Multimonitor:
/*
/*
#include <display.h>
nputchar(r, c, ch)
    int      r, c, ch;
{
    Nxt_img_ch[r][c] = ch;
}

/* nputs.c
/* AND1091 LCD display routine for use with the Multimonitor:
/*
/*
#include <display.h>
nputs(s)
    char      *s;
{
    while (*s) {
        nputc(*s);
        s++;
    }
}

/* nsetattrpos.c:
/* AND1091 LCD display routine for use with the Multimonitor:
/*
/*
#include <display.h>
nsetattrpos(r, c)
    int      r, c;

```

displib.c

...nputc

*/

nputchar

*/

nputs

*/

nsetattrpos

displib.c

```
{
    Nxt_text_row = r;
    Nxt_text_col = c;
}
/...../
/* nsetattrpos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
nsetattr(attr)
    int attr;
{
    Nxt_text_attr = attr;
}
/...../
/* nsetcurspos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
nsetcurspos(r, c)
    int r, c;
{
    Nxt_curs_row = r;
    Nxt_curs_col = c;
}
/...../
/* nsettextpos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
nsettextpos(r, c)
    int r, c;
{
    Nxt_text_row = r;
    Nxt_text_col = c;
}
/...../
```

displib.c

...nsetattrpos

nsetattr

nsetcurspos

nsettextpos

displib.c

```
/* wcln.c:
/* AND1091 LCD display routine for use with the Multimonitor:
/*****

#include <display.h>
wcln(line)
{
    int line;

    int i;

    wsettextpos(line, 0);

    for (i = 0; i < DISP_COLS; i++)
        wputc(' ');

    wsettextpos(0, 0);
}
/*****

/* wcls.c:
/* AND1091 LCD display routine for use with the Multimonitor:
/*****

#include <display.h>
wcls()
{
    int i, j;

    for (i = 0; i < WIN_ROWS; i++) {
        wsettextpos(i, 0);

        for (j = 0; j < DISP_COLS; j++)
            wputc(' ');

        wsettextpos(0, 0);
    }
}
/*****

/* wgetorg.c:
/* AND1091 LCD display routine for use with the Multimonitor:
/*****

#include <display.h>
wgetorg()
{
    return (Win_row_org);
}
```

displib.c

wcln

wcls

wgetorg

displib.c

```

/*****
/* win_to_nxt.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

```

#include <display.h>
win_to_nxt()
{

```

```

    int          i, j, nrow, wrow;

    nrow = Win_row_dorg;
    wrow = Win_row_org;
    for (i = 0; i < Win_row_len; i++, nrow++, wrow++) {
        for (j = 0; j < DISP_COLS; j++) {
            nputchar(nrow, j, Win_ch[wrow][j]);
            nputattr(nrow, j, Win_at[wrow][j]);
        }
    }
    nsettextpos(0, 0);
}

```

```

/*****
/* win_to_nln.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

```

#include <display.h>
win_to_nln(wrow)
{
    int          wrow,

```

```

    int          j, nrow;
    int          offset;

    offset = wrow - Win_row_org;
    if (offset >= Win_row_len || offset < 0)
        return;

    nrow = Win_row_dorg + offset;
    for (j = 0; j < DISP_COLS; j++) {
        nputchar(nrow, j, Win_ch[wrow][j]);
        nputattr(nrow, j, Win_at[wrow][j]);
    }
}

```

```

}

```

displib.c

```

*/

```

win_to_nxt

```

*/

```

win_to_nln

displib.c

```

/*****
/* wmake.c:
/* AND1091 LCD display routine for use with the Multimonitor: */
*****/

#include <display.h>
wmake(drow, len)
    int          drow, len;
{
    Win_row_len = len;
    Win_row_org = 0;
    Win_row_dorg = drow;
    wsetattr(DISP_NORM);
}

/*****
/* wmessage.c:
/* AND1091 LCD display routine for use with the Multimonitor: */
*****/

#include <display.h>
wmessage(r, c, s)
    int          r, c;
    char         *s;
{
    wsettextpos(r, c);
    wputs(s);
}

/*****
/* wputattr.c:
/* AND1091 LCD display routine for use with the Multimonitor: */
*****/

#include <display.h>
wputattr(r, c, attr)
    int          r, c, attr;
{
    Win_at[r][c] = attr;
}

/*****
/* wputc.c:

```

displib.c

*/

wmake

*/

wmessage

*/

wputattr

*/

displib.c

```

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

#include <display.h>
wputc(c)
    int      c;

{

    if (c == '\n') {
        Win_text_col = 0;
        Win_text_row = (Win_text_row + 1) % WIN_ROWS;
    }
    else {
        /* write char */
        wsettextpos(Win_text_row, Win_text_col);

        /* write char's attribute */
        wsetattrpos(Win_text_row, Win_text_col);

        Win_ch[Win_text_row][Win_text_col] = c;
        Win_at[Win_text_row][Win_text_col] = Win_text_attr;

        /* update row and col variables */
        Win_text_col++;

        if (Win_text_col >= DISP_COLS) {
            Win_text_col = 0;
            Win_text_row++;

            if (Win_text_row >= WIN_ROWS) {
                Win_text_row = 0;
            }
        }
    }
}
/*****

/* wputchar c */

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

#include <display.h>
wputchar(r, c, ch)
    int      r, c, ch;
{
    Win_ch[r][c] = ch;
}

```

displib.c

wputc

wputchar

displib.c

```
/*
 * *****
 * wputs.c:
 * AND1091 LCD display routine for use with the Multimonitor:
 * *****
 */
```

```
#include <display.h>
wputs(s)
char *s;
```

wputs

```
{
    while (*s) {
        wputc(*s);
        s++;
    }
}
/* ***** */
```

```
/* wscrollup.c:
 * AND1091 LCD display routine for use with the Multimonitor:
 * *****
 */
```

```
#include <display.h>
wscrollup(n)
int n;
```

wscrollup

```
{
    Win_row_org -= n;
    if (Win_row_org < 0) {
        Win_row_org = 0;
    }
    win_to_next();
    dupdte();
}
/* ***** */
```

```
/* wscrollup.c:
 * AND1091 LCD display routine for use with the Multimonitor:
 * *****
 */
```

```
#include <display.h>
wscrollup(n)
int n;
```

wscrollup

```
{
    int lastpos;

    lastpos = WIN_ROWS - Win_row_len - 1;
```

displib.c

```
Win_row_org += n;
if (Win_row_org > lastpos) {
    Win_row_org = lastpos;
}
win_to_next();
dupdate();
}
/*****
/* wsetattrpos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsetattrpos(r, c)
    int          r, c;
{
    Win_text_row = r;
    Win_text_col = c;
}
/*****
/* wsetattr.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsetattr(attr)
    int          attr;
{
    Win_text_attr = attr;
}

/*****
/* wsetcurspos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsetcurspos(r, c)
    int          r, c;
{
    Win_curs_row = r;
    Win_curs_col = c;
}
```

displib.c

...wscrollup

*/

wsetattrpos

*/

wsetattr

*/

wsetcurspos

displib.c

```
}
/...../

/* wsettextpos.c
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
wsettextpos(r, c)
{
    int r, c;

    Win_text_row = r;
    Win_text_col = c;
}
/...../

/* wupdate.c:
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
wupdate()
{
    win_to_nxt();
    dupdate();
}

/...../

/* wupdl.c:
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
wupdl(i)
{
    int i;

    int j;

    if (i < Win_row_org || i >= Win_row_len + Win_row_org)
        return;

    wla_to_nla(i);
    j = Win_row_dorg + i - Win_row_org;
    dupdl(j);
}
```

displib.c

...wsetcurspos

*/

wsettextpos

*/

wupdate

*/

wupdl

--
displib.c

--
displib.c

DRAWINGS, ENGINEERING AND ASSOCIATED LISTS
LEVEL 1

FOR THE

PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A007

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

DATA W.A. HILLENBRAND BIOMED.
LIST ENGR. CENTER, PURDUE UNIV.

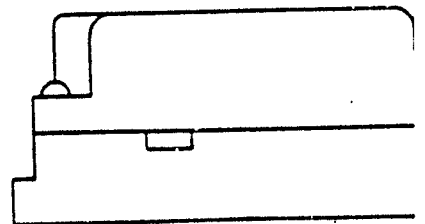
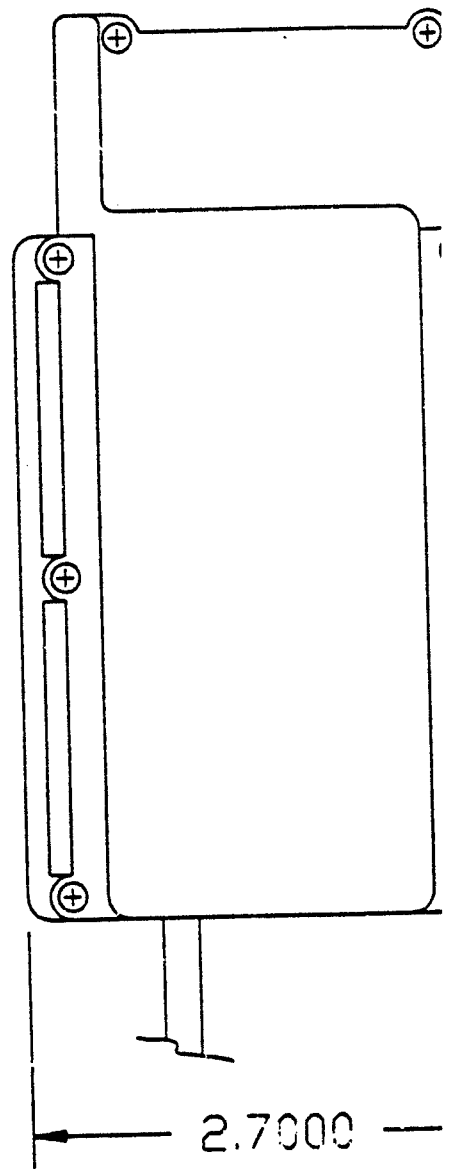
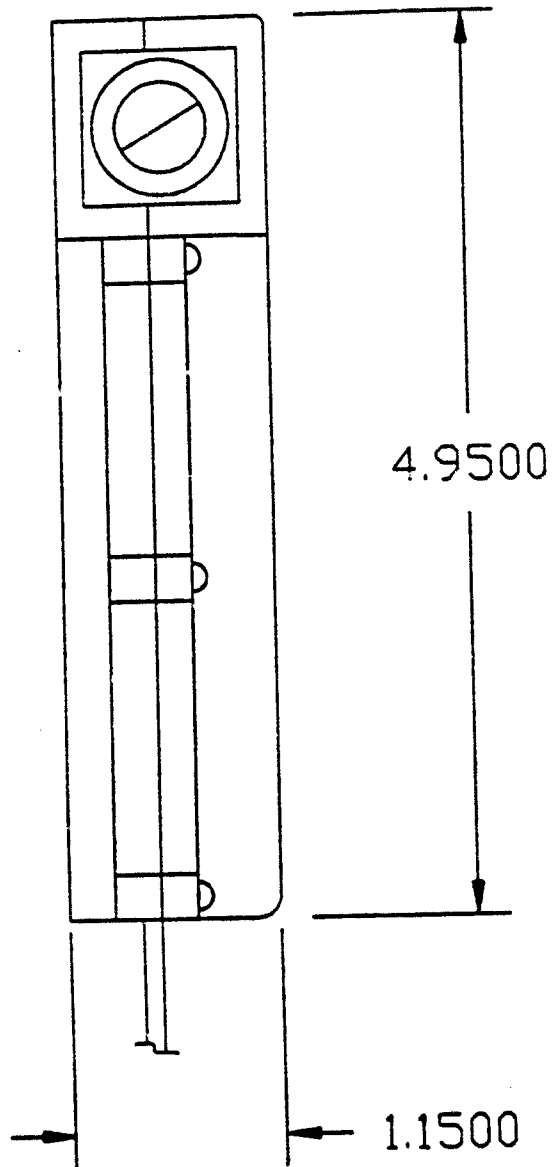
DAMD17-87 C-7195

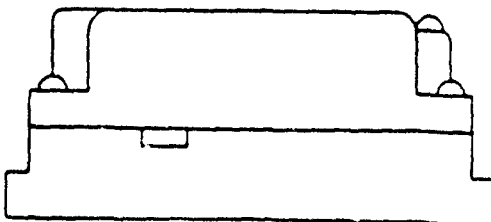
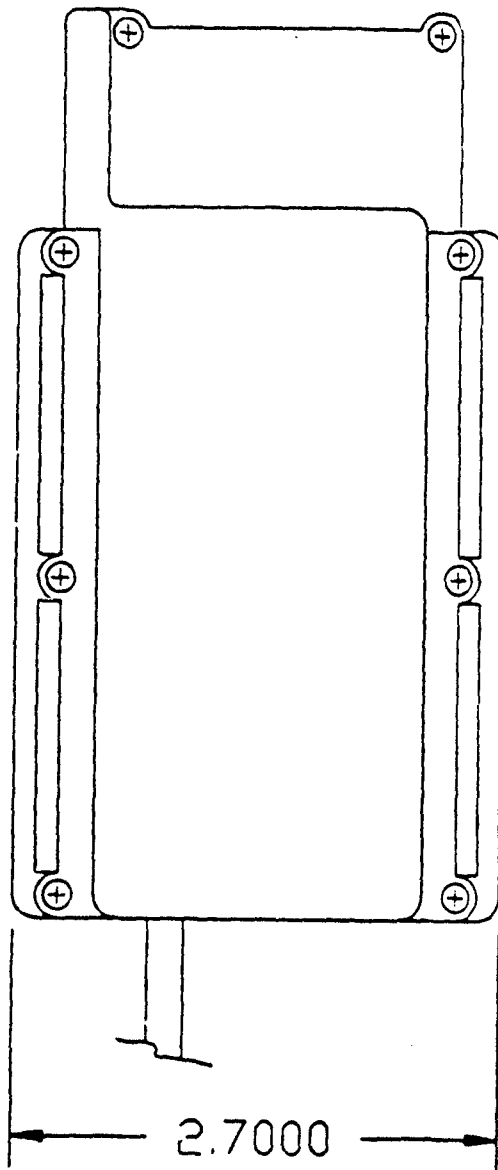
A007-DL 9/30/89

DRAWINGS, ENGINEERING & ASSOCIATED LISTS

SHEET 1 OF 1

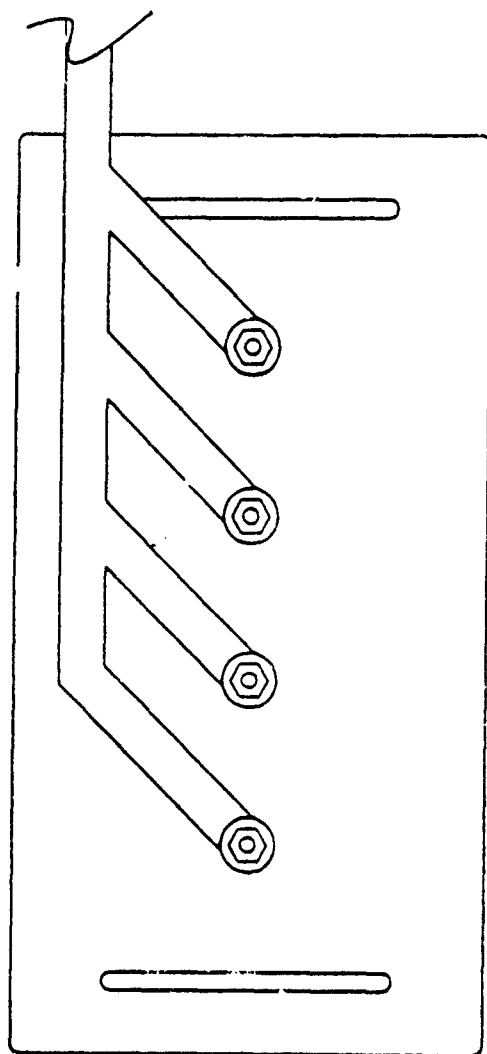
DWG SIZE	DOCUMENT NUMBER	SHEET NUMBER	NOMENCLATURE OR DESCRIPTION
B	PMC-MEC-ASY	1-2	PMC OUTLINE DIMENSIONS
A	PMC-ELE-ASY	1 OF 5	PMC ASSEMBLY - BOTTOM VIEW
A	PMC-ELE-ASY	2 OF 5	PMC ASSEMBLY - EDGE VIEW
A	PMC-ELE-ASY	3 OF 5	PMC ASSEMBLY - TOP VIEW
A	PMC-ELE-ASY	4 OF 5	PMC CASE BOTTOM
A	PMC-ELE-ASY	5 OF 5	PMC ASSEMBLY - TOP REMOVED
A	PMC-ASY-PL	1 OF 1	PMC PARTS LIST
B	HHM-MEC-ASY	1 OF 1	HHM OUTLINE DIMENSIONS
A	HHM-ELE-ASY	1 OF 3	HHM FRONT PANEL REMOVED
A	HHM-ELE-ASY	2 OF 3	HHM REAR PANEL REMOVED
A	HHM-ELE-ASY	3 OF 3	HHM FRONT PANEL - REAR VIEW
A	HHM-ASY-PL	1 OF 1	HHM PARTS LIST
A	PMC-INT-BLK	1 OF 1	PMC INTERFACE BLOCK DIAGRAM
A	HHM-INT-BLK	1 OF 1	HHM INTERFACE BLOCK DIAGRAM
A	PMC-FLO-PRE	1 OF 1	PMC PREPROCESSOR FLOW CHART
A	PMC-FLO-SGP	1-2	PMC SIGNAL PROCESSING FLOW CHART



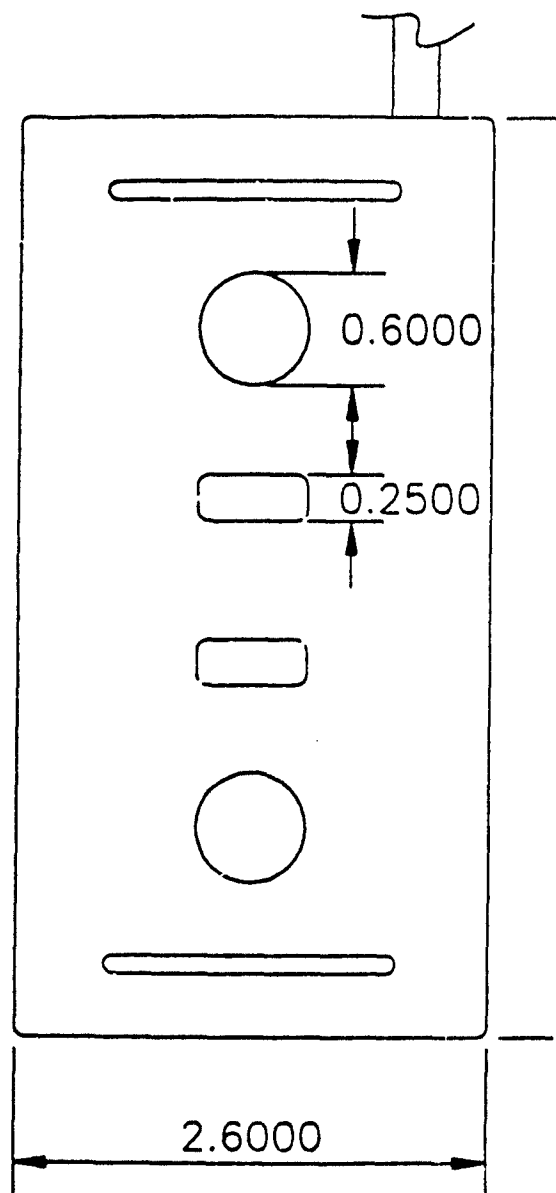


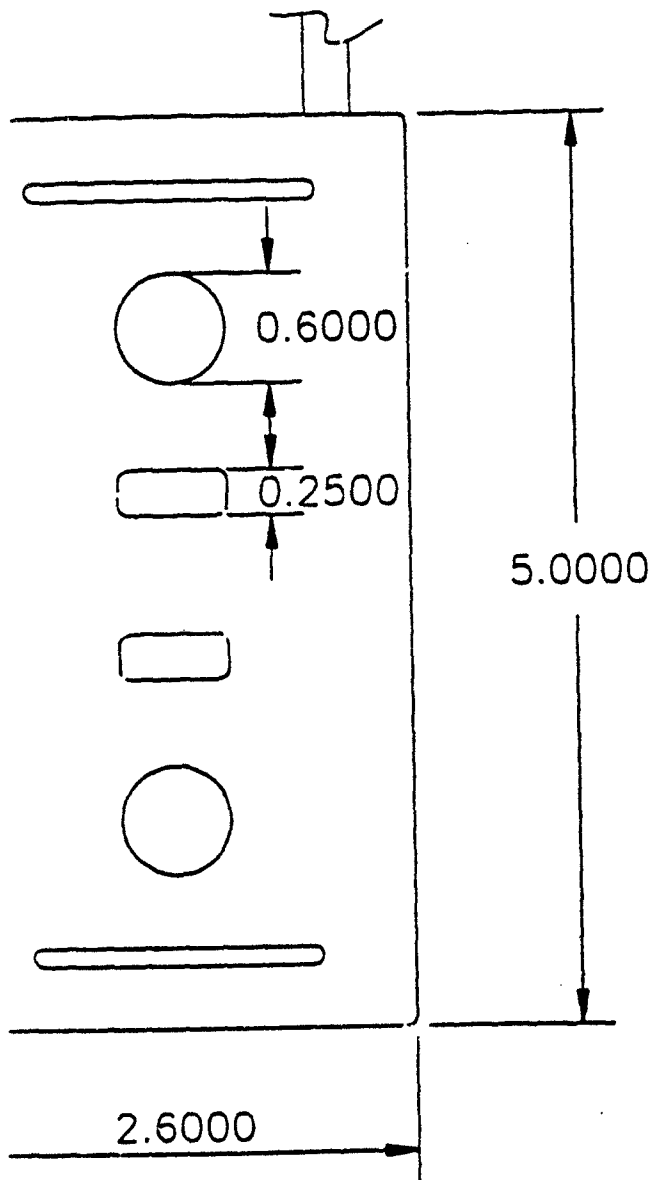
CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDIC. WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PPC 1	
DRAWN <i>RSF</i>	7/12/87		
CHECKED <i>RSF</i>	7/12/87		
		SIZE B	FSCM NO.
		SCALE 1/1	

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PPC 3.0 DE LINE USION	
DRAWN <i>RSE</i>	<i>1/12/87</i>		
CHECKED <i>RSE</i>	<i>1/12/87</i>		
		SIZE B	FSCM NO.
		DRAWING NO. <i>PPC 3.0 DE LINE USION</i>	
		SCALE 1/1	SHEET 1 OF 2

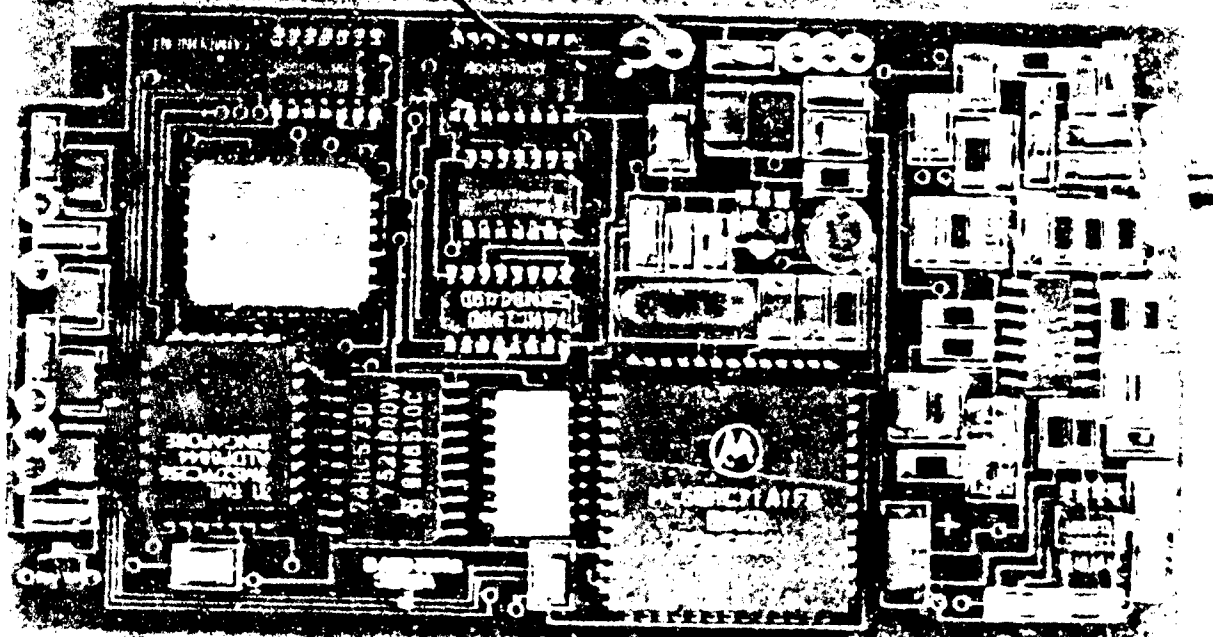


0.9000
0.9000
0.9000

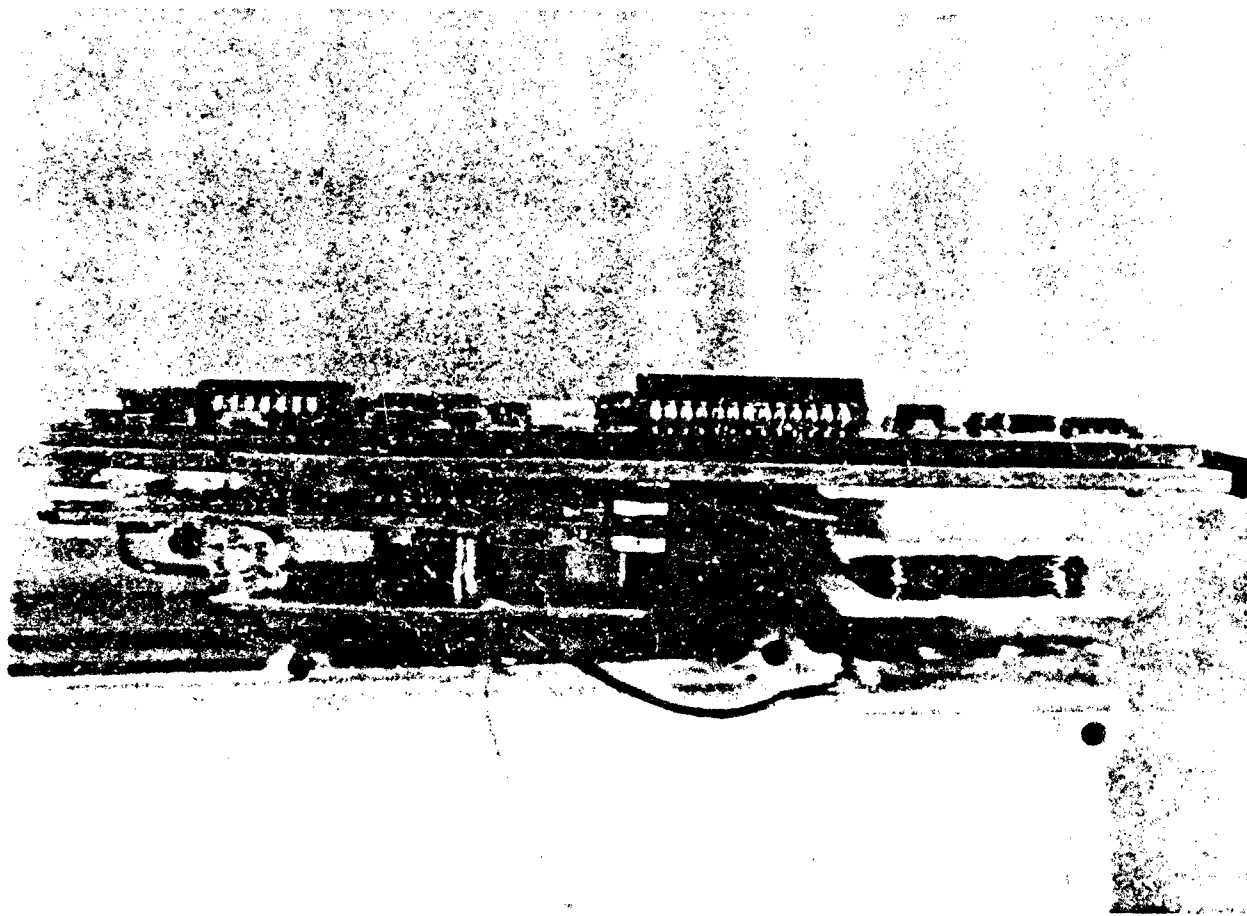




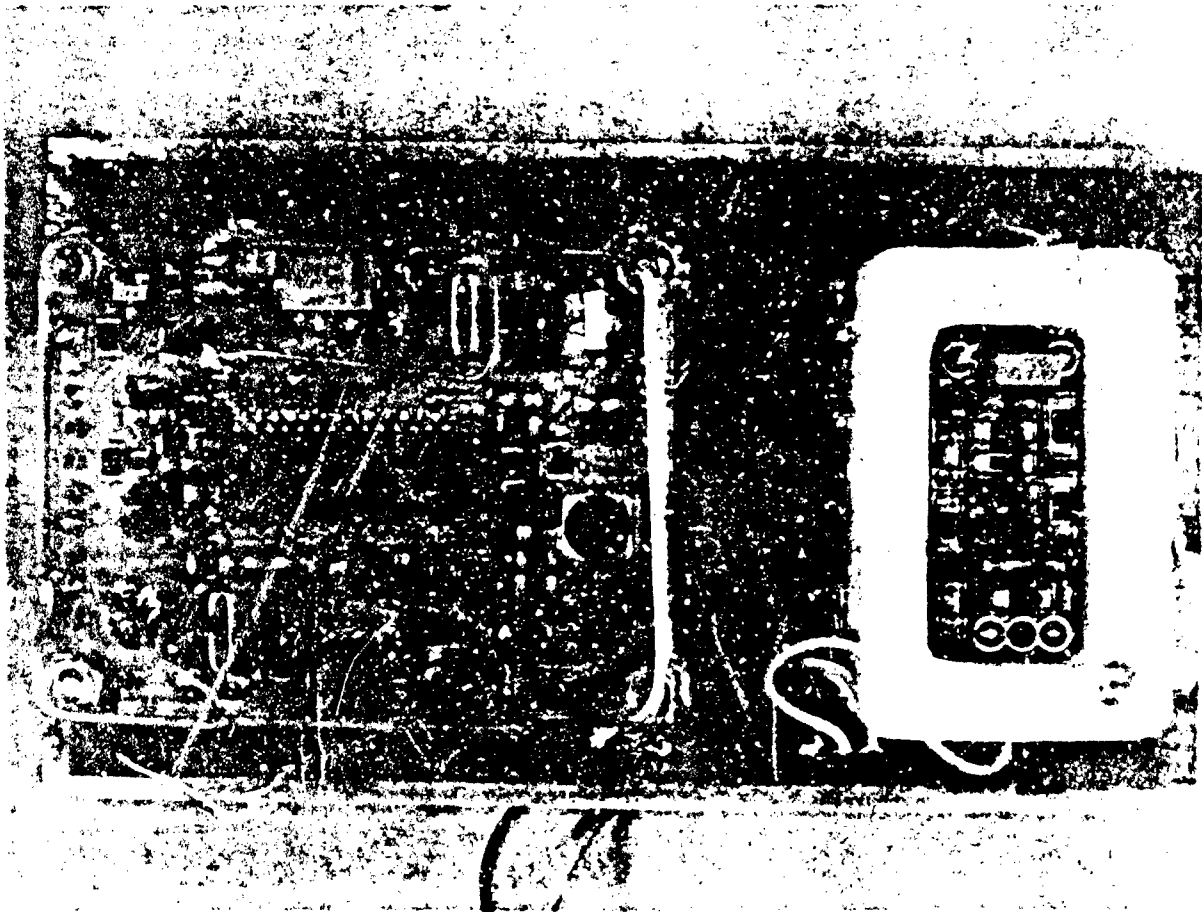
SIZE B	FSCM NO.	DRAWING NO. 174-111-A
SCALE		SHEET 2 OF 2



CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907		
APPROVALS	DATE			
DRAWN	9-10-89			
CHECKED				
		SIZE A	FSCM NO.	DRAWING NO.
		SCALE N/A		SHEET OF

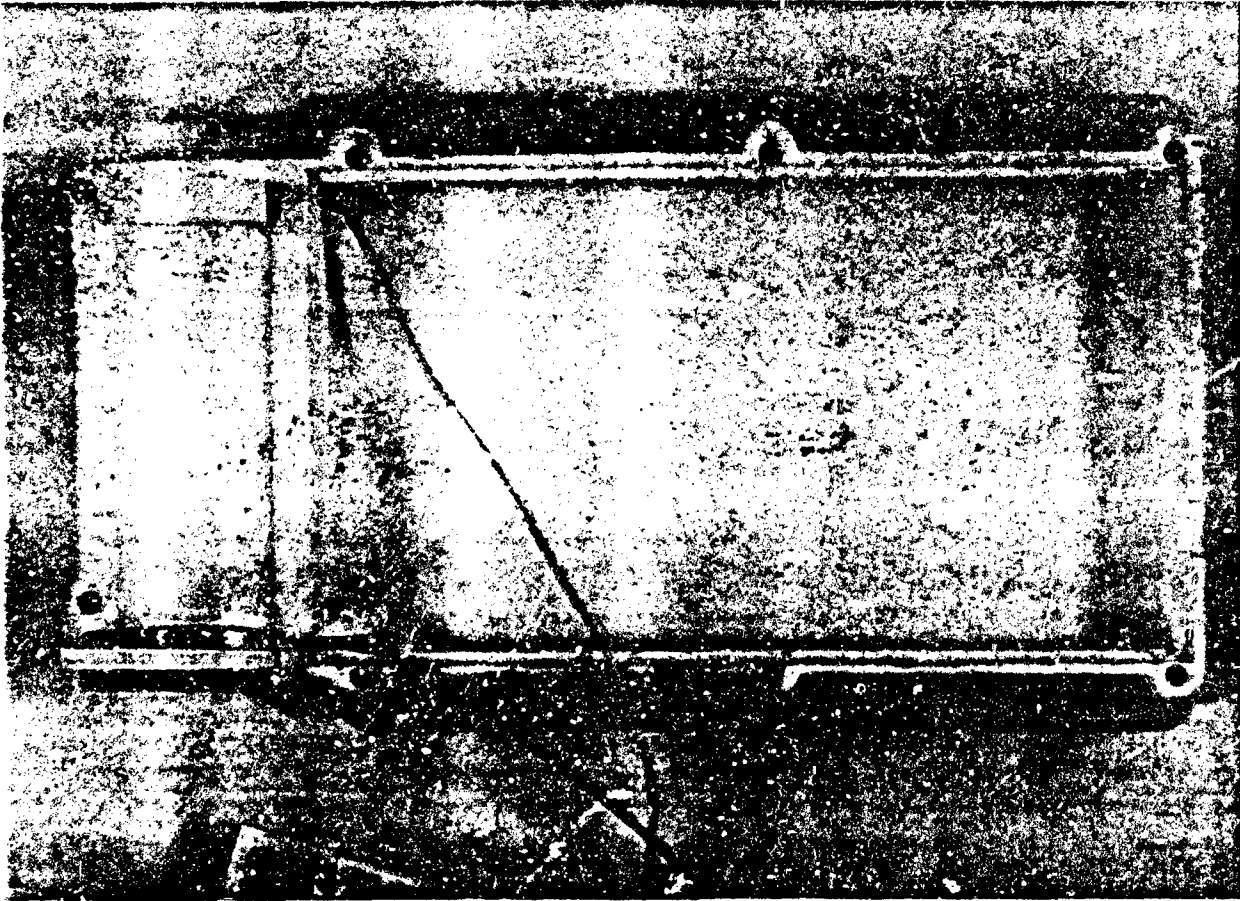


SIZE A	FSCM NO.	DRAWING NO.	
SCALE		SHEET	OF



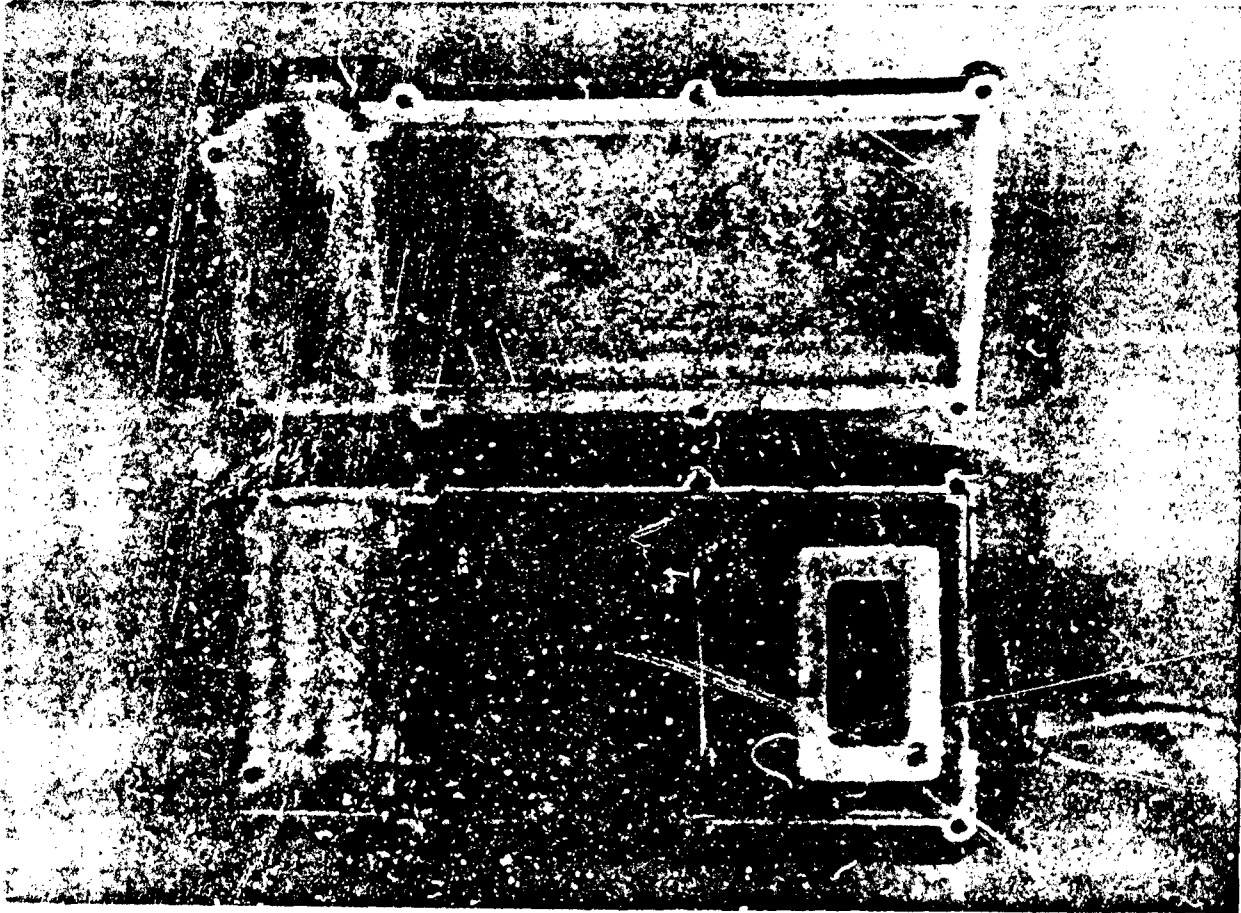
Top View

SIZE A	FSCM NO.	DRAWING NO. 412-ELE-AS
SCALE		SHEET 3 OF 5



200-6700-1210

SIZE A	FSCM NO.	DRAWING NO. P.M.C.-ELE-ASV
SCALE		SHEET -- OF 5



Top View - Top Removed

SIZE A	FSCM NO.	DRAWING NO. PMC-ELE-AD
SCALE		SHEET 5 OF 5

PARTS W.A. HILLENBRAND BIOMED. DAMD17-87-C-7195 PMC-ASY-PL 9/30/89
LIST ENGR. CENTER, PURDUE UNIV.

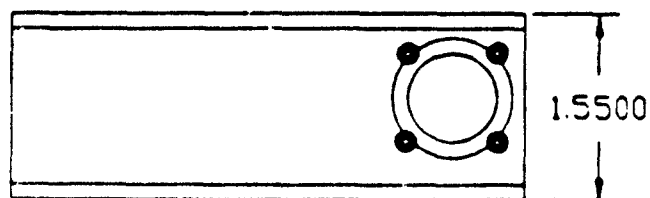
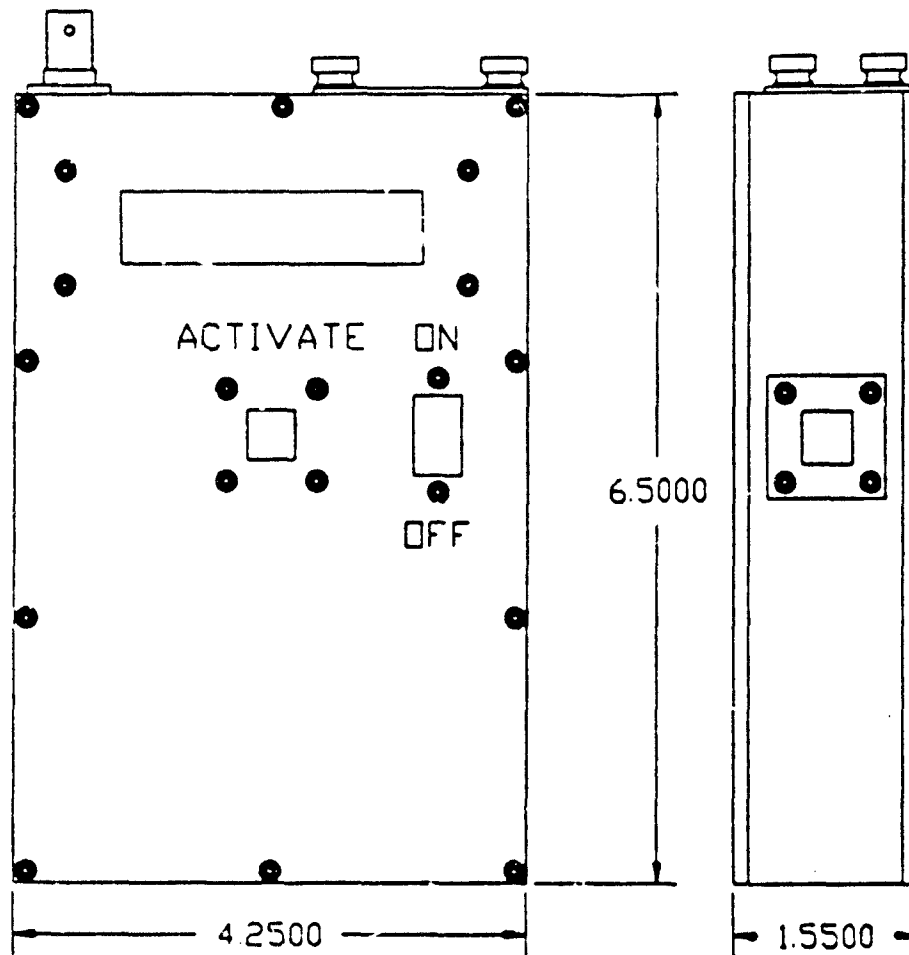
PMC PARTS LIST

SHEET 1 OF 1

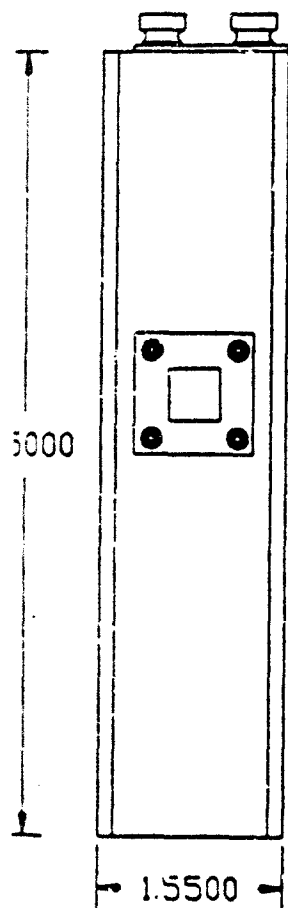
QTY
RQD

NOMENCLATURE OR DESCRIPTION

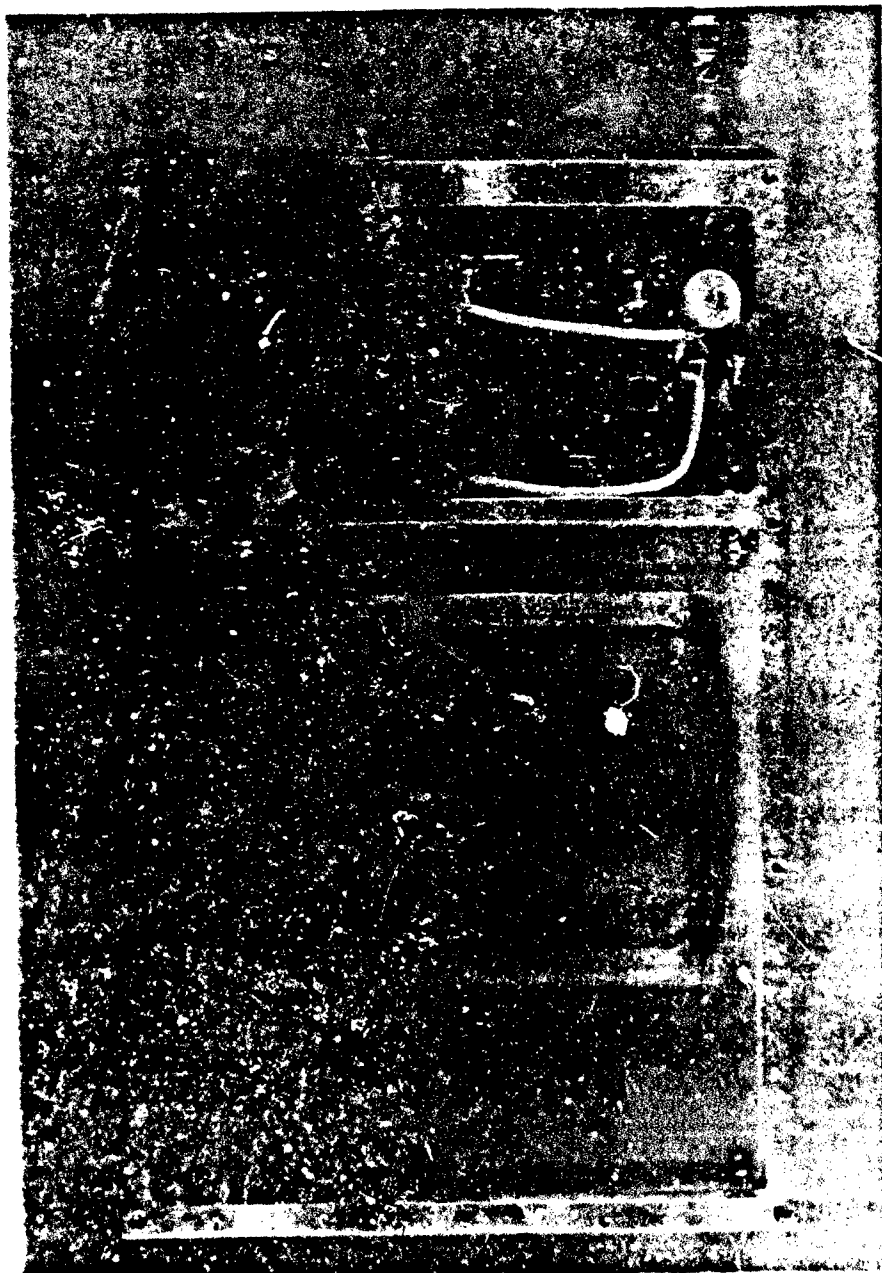
1	case bottom
1	case top
1	battery cap receiver
1	battery cap
1	battery contact assy.
1	wriststrap assy.
1	electronics assy.
6	#2-56 x 0.7 RHP SS
2	#2-56 x 1.0 RHP SS
1	leather strap (2.6x5 in.)
1	wiring harness
3	elastic strap
2	electrodes, .6 in. dia. SS
2	electrodes, .6 x .25 in. SS
4	#4 flat washers
4	#2 lock washers
4	#2 nuts



CONTRACT NO.		PURDUE UNIV	
DAMD17-87-C-7195		W. A. HILLENE	
APPROVALS		DATE	
DRAWN			
CHECKED			
		SIZE	150
		B	
		SCALE	1"

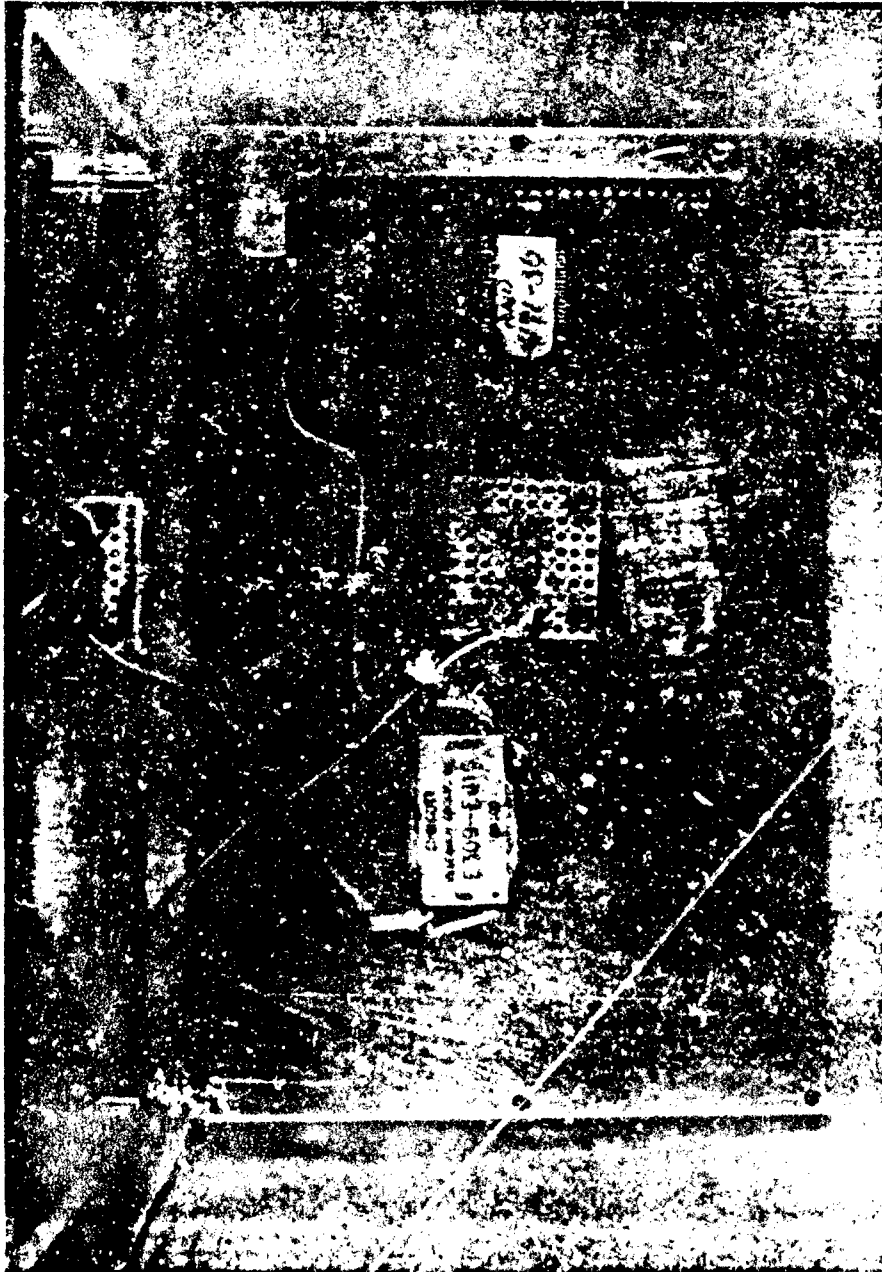


CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE		
DRAWN <i>K. J.</i>	<i>1/15/87</i>		
CHECKED <i>K. J.</i>	<i>1/15/87</i>		
		SIZE B	DRAWING NO. <i>11E1-237</i>
		SCALE <i>1" = 1/2"</i>	SHEET <i>1</i> OF <i>1</i>



1000 PANEL REMOVED

SIZE A	FSCM NO.	DRAWING NO. H-1-E-E-A
SCALE		SHEET OF



QF-78-603-151

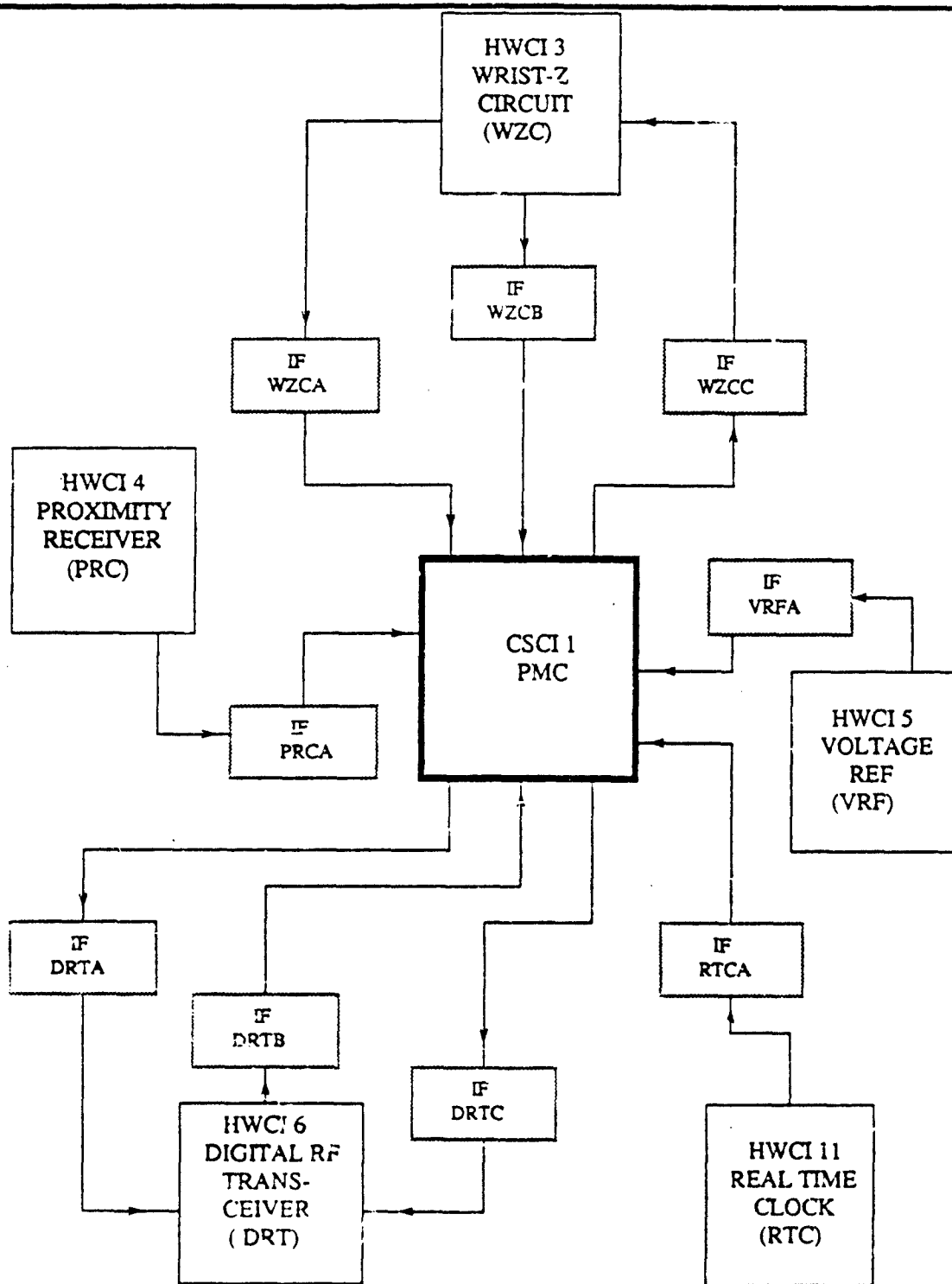
SIZE A	FSCM NO.	DRAWING NO. QF-78-603-151
SCALE		SHEET 1 OF 1

PARTS W.A. HILLENBRAND BIOMED. DAMD17-87-C-7195 HHM-ASY-PL 9/30/89
 LIST ENGR. CENTER, PURDUE UNIV.

HHM PARTS LIST

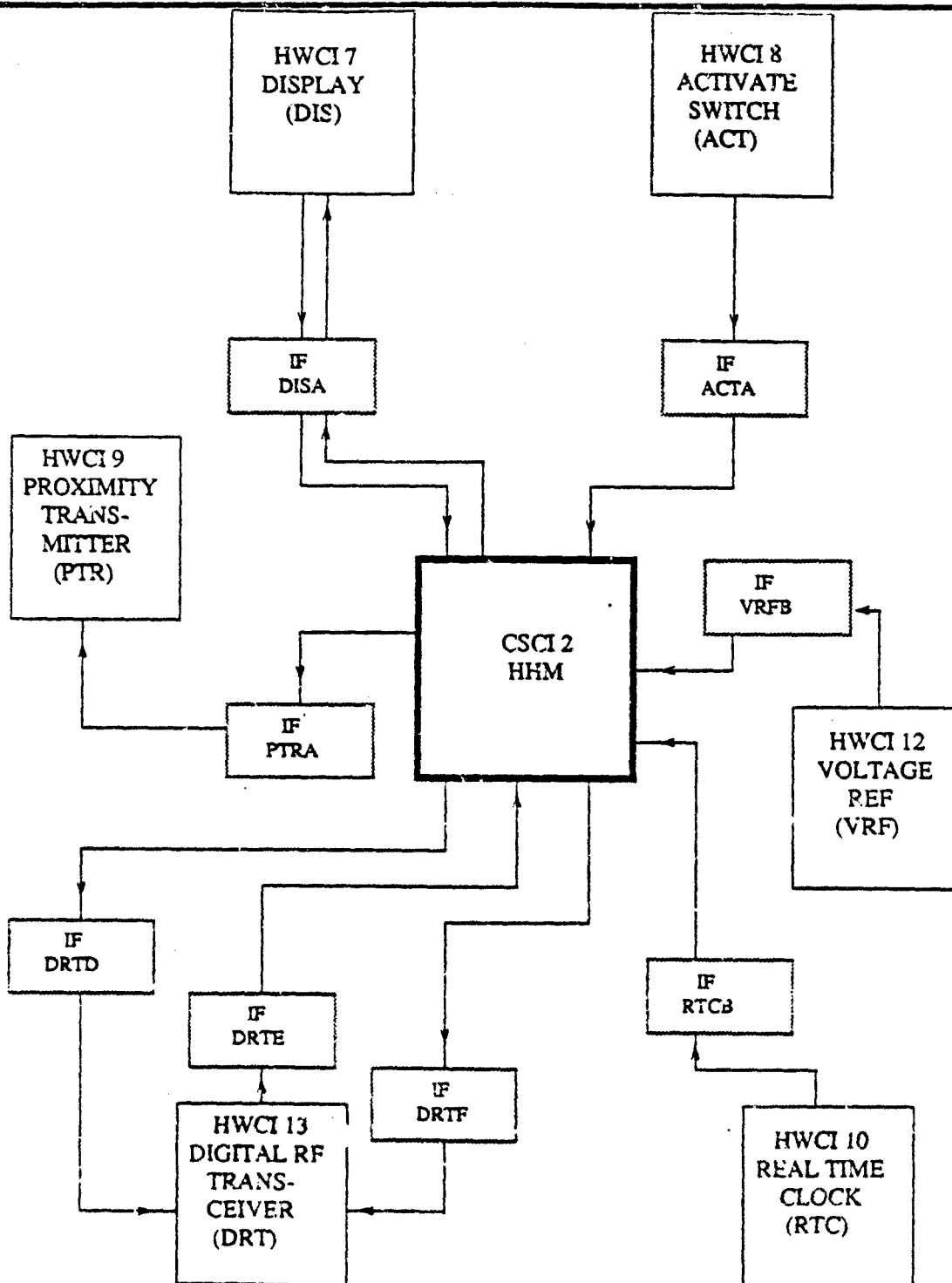
SHEET 1 OF 1

QTY RQD	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION
1		front panel
1		rear panel
1		frame
1		battery cover
1		battery holder
1		battery cap
2	C&K KS11R22CQ	pushbutton switch
2		switch contact board
1		pushbutton cover plate
1		pushbutton spacer
1	C&K 7101J1ZQ	rocker switch
1		prox trans. assy.
1		radio assy.
1		antenna
1		antenna connector
1		CPU board
1	AND491-30	LCD display
1	Durell 04Z	LCD backlight
1	ERG E309-E819	backlight power supply
1		Lexan display window
14		#2-56 x .5 FH socket
30		#2-56 x .25 FH socket
4		#2-56 x .5 RH
4		#2-56 x .25 RH
24		#2-56 nuts
18		#2 flat washers
18		#2 lock washers



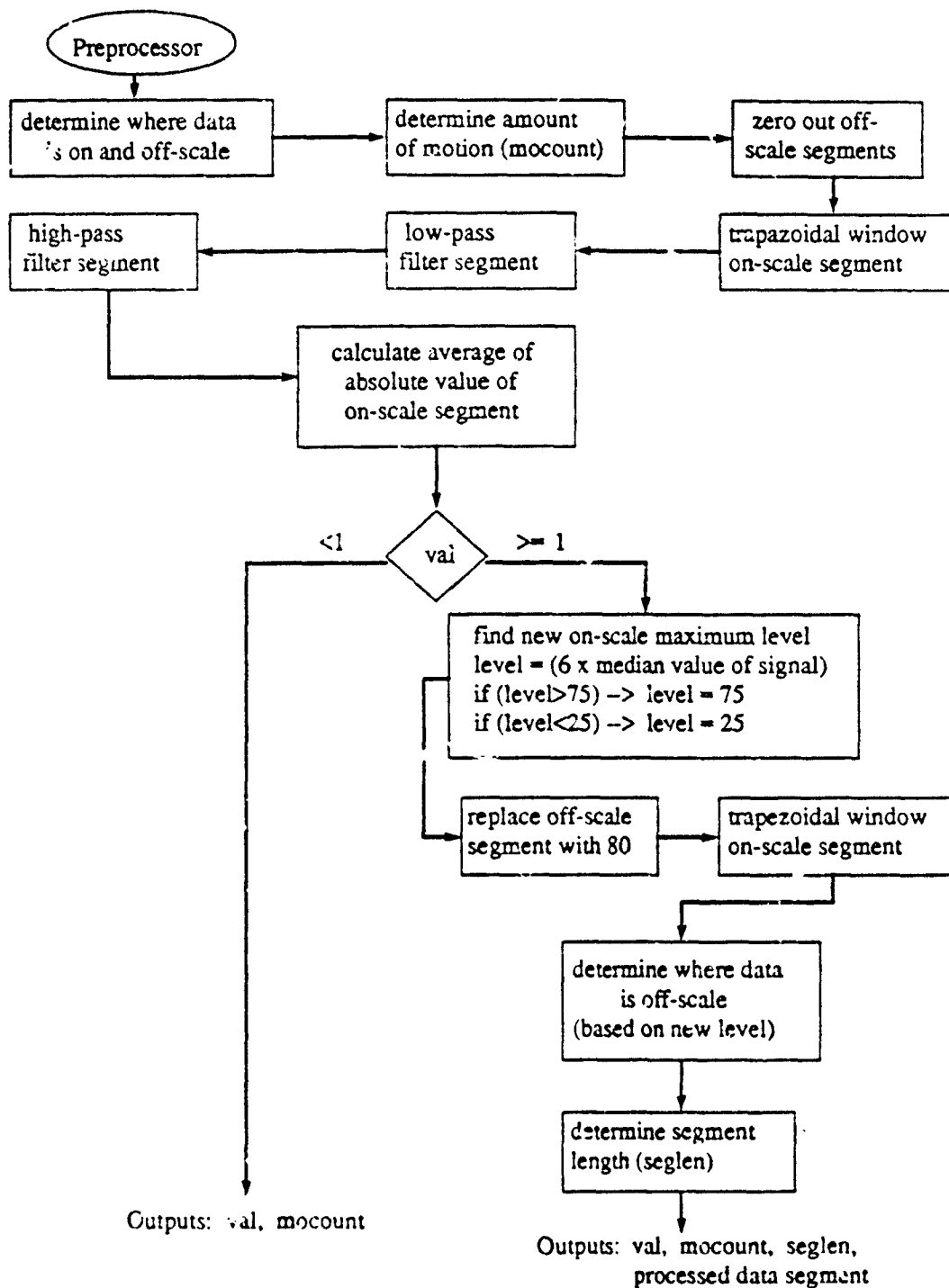
NOTE:
IF = INTERFACE

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC INTERFACE BLOCK DIAGRAM	
DRAWN	9-10-89		
CHECKED	11-6-89		
SIZE	FSCM NO.	DRAWING NO.	
A		PMC-INT-BLK	
SCALE	N/A	SHEET 1 OF 1	

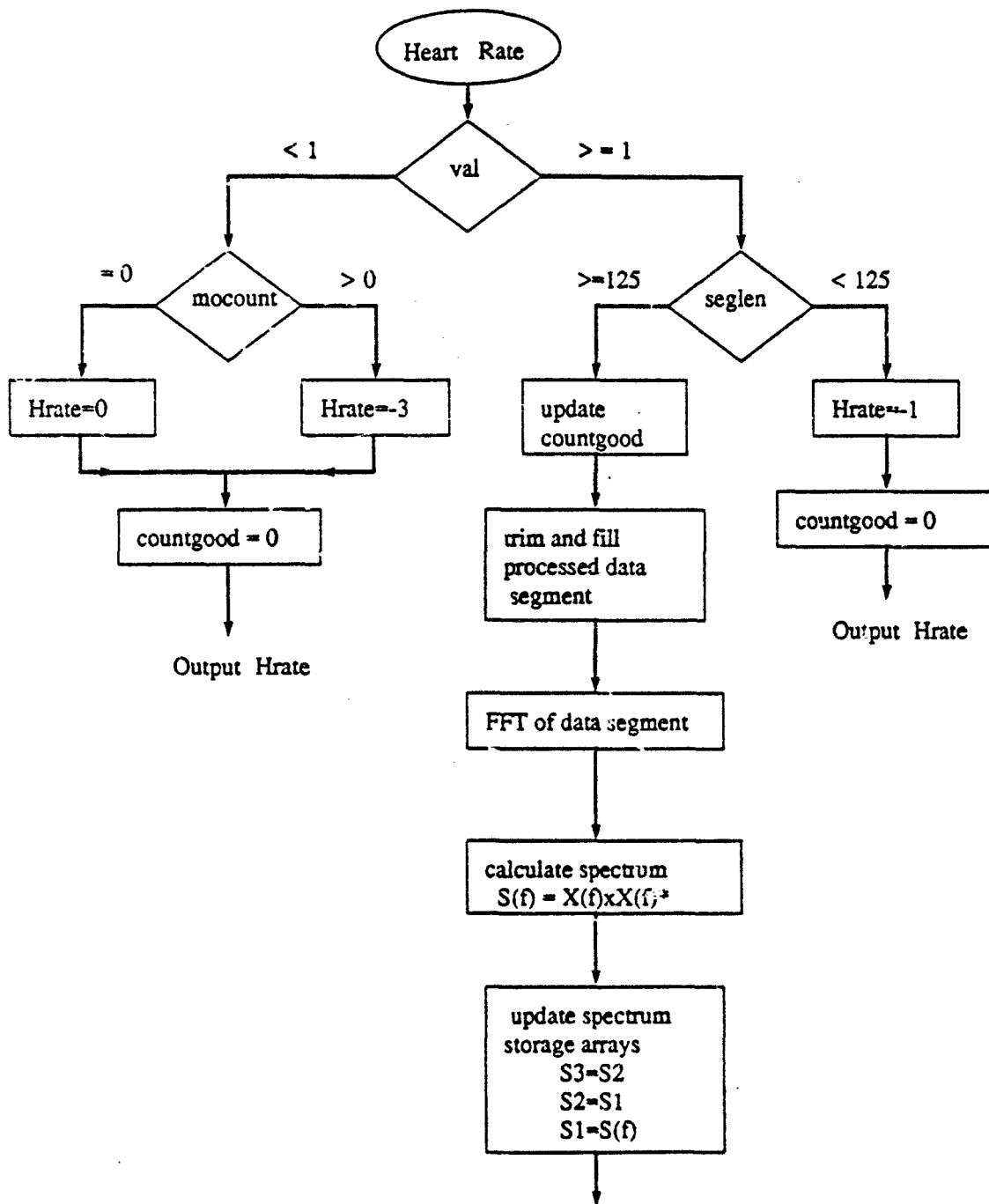


NOTE:
IF = INTERFACE

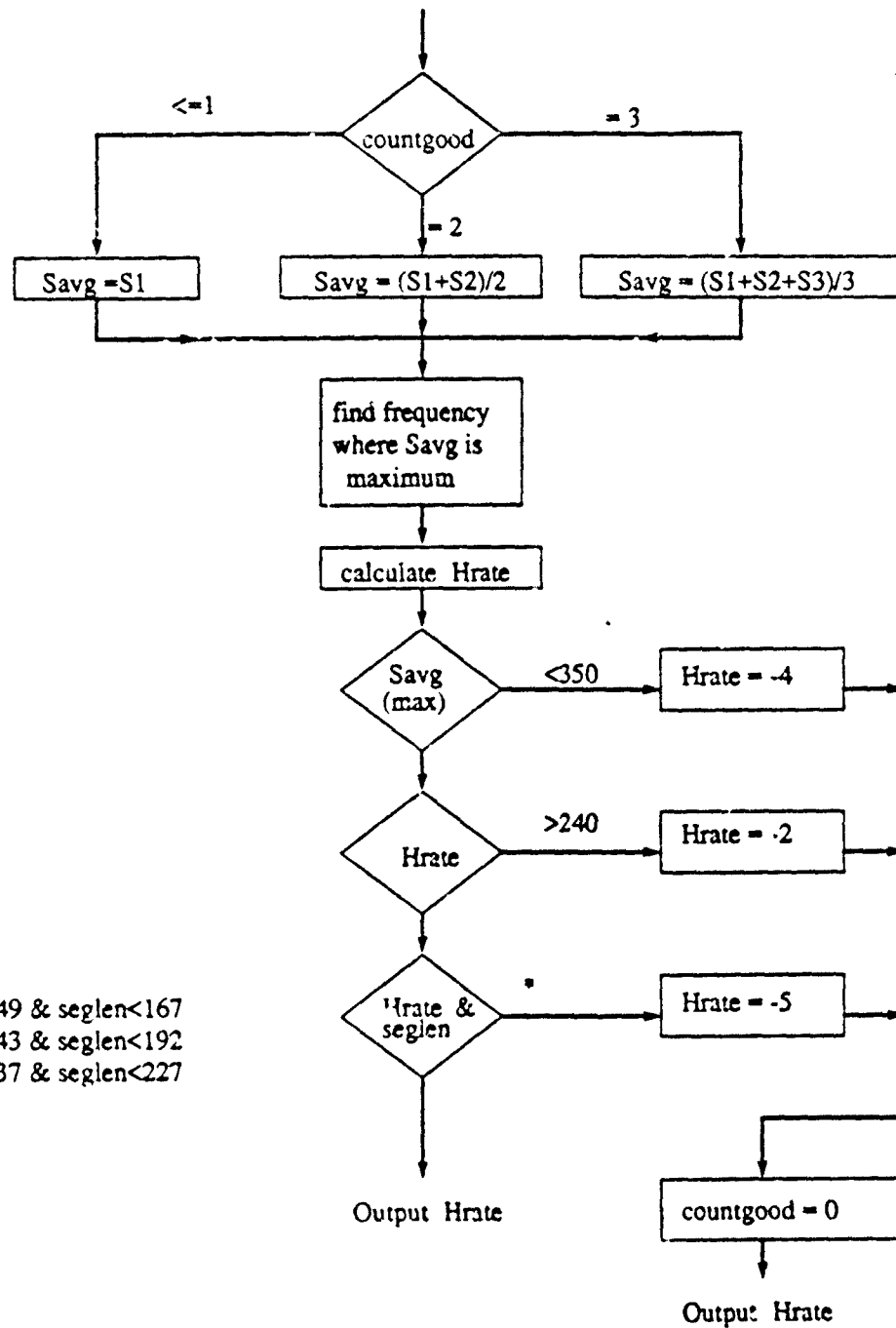
CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	HHM INTERFACE BLOCK DIAGRAM	
DRAWN	9-10-89		
CHECKED S-E	11-6-89		
SIZE	FSCM NO.	DRAWING NO.	
A		HHM-INT-PLK	
SCALE	N/A	SHEET	OF 1



CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC SIGNAL PROCESSING PREPROCESSOR FLOW CHART	
DRAWN U.P.	9-10-89		
CHECKED JTS	11-6-89	DRAWING NO. PMC-FLO-FRE	
		SIZE A	FSC# NO.
		SCALE N/A	SHEET 1 OF 1



CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC SIGNAL PROCESSING TECHNICAL REPORT	
DRAWN <i>u.f.</i>	9-10-89		
CHECKED <i>---</i>	<i>---</i>	SIZE A	FSCM NO.
		DRAWING NO. PMC-FLO-SGP	
		SCALE N/A	SHEET 1 OF 2



• $Hrate < 49$ & $seglen < 167$
 • $Hrate < 43$ & $seglen < 192$
 • $Hrate < 37$ & $seglen < 227$

SIZE A	FSO. NO.	DRAWING NO. PMC-FLO-SGD
SCALE 1/16		SHEET 3 OF 2

DATA AND DRAWING PACKAGE
FOR ELECTRONIC EQUIPMENT

FOR THE

PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A008

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

DATA W. A. HILLENBRAND BIOMED. DAMD17-87-C-7195 A008-DL 9/10/89
LIST ENGR. CENTER, PURDUE UNIV.

DATA AND DRAWING PACKAGE FOR ELECTRONIC EQUIPMENT SHEET 1 OF 1

DWG SIZE	DOCUMENT NUMBER	SHEET NUMBER	DESCRIPTION
A	PMC-RAD-ADJ	1	PMC RADIO ADJUSTMENT PROCEDURE
A	PMC-SYS-TST	1-2	PMC SYSTEM BENCH AND FUNCTIONAL TEST
B	PMC-CPU-PWM	1	PMC MAIN BOARD COMPONENT LAYOUT
B	PMC-RAD-PWM	1-2	PMC RADIO LINK PRINTED WIRING MASTER
B	HHM-RAD-PWM	1	HHM RADIO LINK PRINTED WIRING MASTER

Magnavox Radio Adjustment Procedure
Models 801 and 802

Equipment

RF generator with 72.168 MHZ output and 39" stick antenna.
100 MHZ bandwidth oscilloscope.
Nonconductive screwdriver.
RF field strength meter or spectrum analyzer with antenna.

1. Connect RF generator to antenna and set for 72.168 MHZ, FM modulation with 1KHZ tone, 4KHZ deviation, and 0.5 Vrms into the stick antenna
2. Connect oscilloscope to the RECEIVE AUDIO pin (P1-6) of the radio module.
3. Adjust L3 for best symmetry of the 1KHZ sine wave.
4. Connect oscilloscope to pin 1 of U1 on the radio module.
5. Reduce RF generator to 50mV rms into the antenna.
6. Adjust C5 for maximum amplitude at U1 pin 1.
7. Disconnect oscilloscope from the radio module.
8. With the radio powered from batteries (6V) and with NO GROUND attached, connect the KEY line (P1-3) to the negative (-) terminal of the battery supply.

FOR MODEL 801

9. Monitor the radio transmitter output with the spectrum analyzer or field strength meter. Adjust C8 for maximum transmitter output power.

FOR MODEL 802

10. Same as 9 above except adjust C29 and the series antenna capacitor.

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC RADIO ADJUSTMENT PROCEDURE	
DRAWN GG	9-10-89		
CHECKED JTI	11-1-89		
		SIZE A	DRAWING NO. PMC-RAD-ADJ
		SCALE N/A	SHEET 1 OF 1

10. INTERFACE TEST SUMMARY

10.1 *Scope.* This appendix establishes the test procedure used to test the interface between the PMC units and the HHM units.

10.2 *Bench Test.*

10.2.1 *Overview of Bench Test.* The serial communication signal lines of a PMC and a HHM were connected directly together. This connection served to bypass the radio transceivers. Temporary pushbutton switches were attached to the PMC proximity interrupt and the HHM activate. Three tests were then performed on the hard wired units.

10.2.2 *General Results of Bench Test.*

10.2.2.1 *Supply Current Test.* A DC millammeter was connected in series with the PMC power supply. The power supply was turned on and the supply current was measured and recorded. The PMC current value was approximately 21 mA. After waiting for one minute, the supply current was again measured and recorded. At this time the PMC CSCI entered the "STOP" state and the PMC current reading was approximately 0.6 mA.

10.2.2.2 *PMC Activation Test.* Actual PMC activation was simulated by pressing the HHM activate pushbutton and the PMC proximity interrupt pushbutton in quick succession. An oscilloscope monitoring the HHM proximity signal output line displayed a burst of 125 KHz square waves when the HHM was activated. The HHM display read

PMC # Activated

where # was the identification number of the PMC. Monitoring the PMC radio power line with an oscilloscope showed the line became active when the PMC was activated.

10.2.2.3 *Status Test.* The HHM would query the activated PMC every 15 seconds. Initially the HHM would receive the message

Off Wrist

The oscilloscope monitoring the radio key line verified proper operation of the line for both the HHM and PMC. The PMC wrist strap was then applied to the wrist

CONTRACT NO. DAMD17-87-G-7195		PURDUE UNIVERSITY W. A. MILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC SYSTEM BENCH AND FUNCTIONAL TEST	
DRAWN	9-10-89		
CHECKED			
		SIZE A	DRAWING NO. PMC-SYS-TST
		SCALE N/A	SHEET 1 OF 2

of a subject. The subject remained still while the HHM display was monitored. The subject's heart rate was obtained from the subject's pulse. This heart rate was compared to the results displayed by the HHM. The last step was for the subject to intentionally move during data collection. During this time the HHM display reported motion.

10.3 Functional Test.

10.3.1 *Overview of Functional Test.* This test checked the performance and the accuracy of the various PMCs and HHMs. The units were tested by strapping a PMC on a subject and recording the subject's heart rate or motion reading which was displayed by the HHM every 15 seconds. The HHM heart rate was compared with the subject's actual heart rate which was obtained by feeling the subject's pulse. Each PMC was tested continuously for a period of 90 minutes. The testing occurred while the subject was resting, walking, resting after walking, running, and resting after running. All delivered PMCs and HHMs were tested.

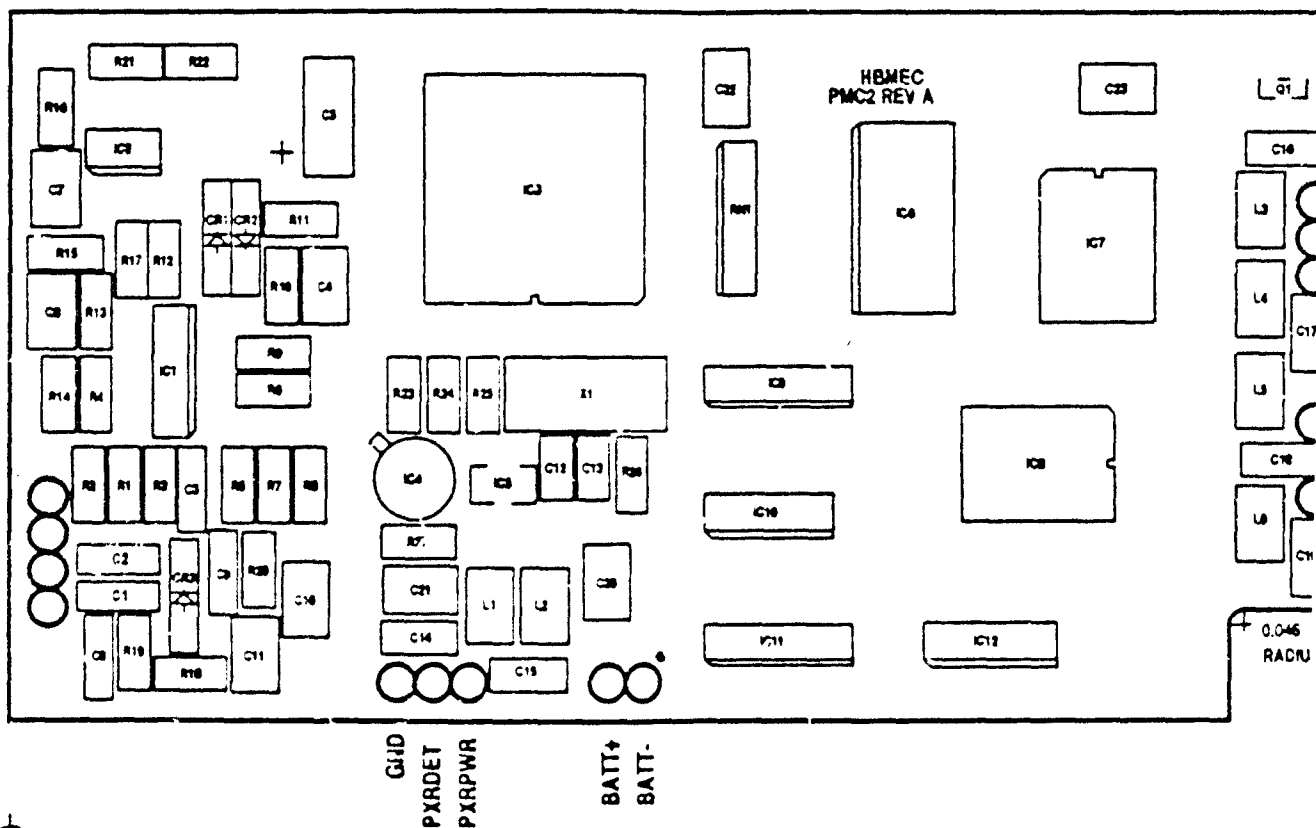
10.3.2 *General Results of Functional Test.* The PMCs and HHMs generally performed as expected. Accurate heart rates were obtained when the subject was resting and when the subject was resting after walking. Some heart rates transmitted during light walking were accurate. Most motion caused the HHM to display a motion reading rather than a heart rate value. Running produced the

Heavy Motion

message to appear. Following exercise, the PMC occasionally transmitted heart rates less than 60 bpm where heart rates in the range of 120 bpm were expected. Such readings probably resulted by the impedance signals due to the heavy breathing mixing with the impedance signals due to the heart pumping blood. The HHM had no problem in obtaining information from the PMC.

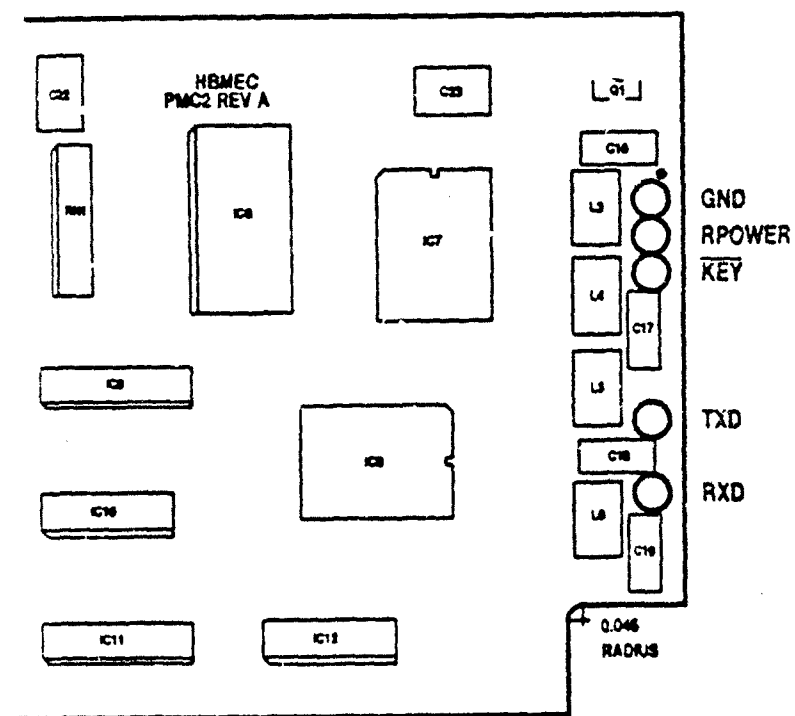
SIZE A	FSCM NO.	DRAWING NO. PMC-SYS-TST
SCALE N/A		SHEET 1 OF 3

ELECT4
ELECT3
ELECT2
ELECT1



REDUCE TO 3.750 INCHES

CONTRACT NO.	
DAMD17-87-C-7195	
APPROVALS	DATE
DRAWN	10-18-89
CHECKED	

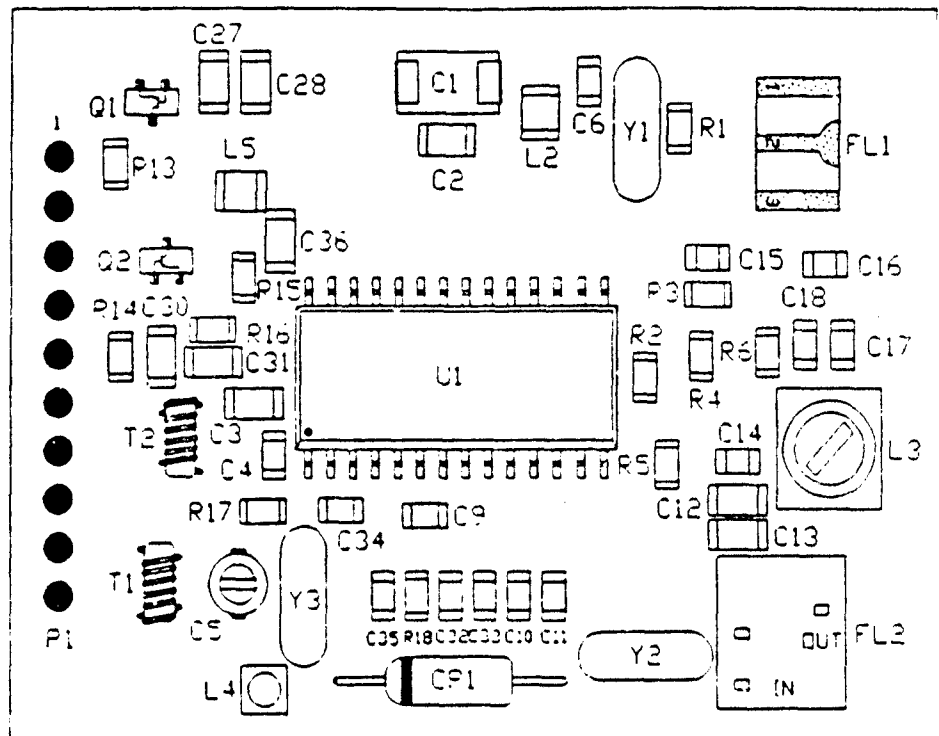


PARTS PLACEMENT

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. MILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC MAIN BOARD PRINTED WIRING MASTER	
DRAWN	10-18-89		
CHECKED			
		SIZE B	FSCM NO.
		DRAWING NO. PMC-CPU-PWM	
		SCALE 2/1	SHEET 1 OF 8

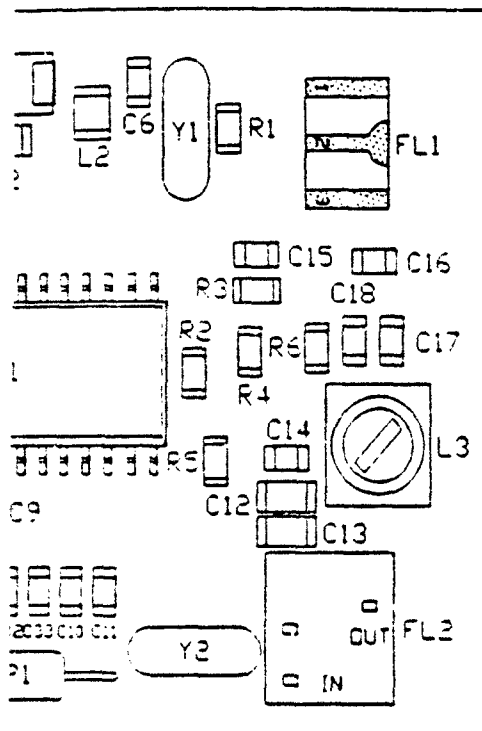
COMPONENT PLACEMENT - (-801 ONLY)

GROUND
 +6VDC
 KEY LINE
 CSAFE
 TX AUDIO
 RX AUDIO
 TX DATA
 +6VDC
 RX DATA
 CARRIER DETECT



CONTRACT NO.
DAMD17-8
APPROVALS
DRAWN
CHECKED

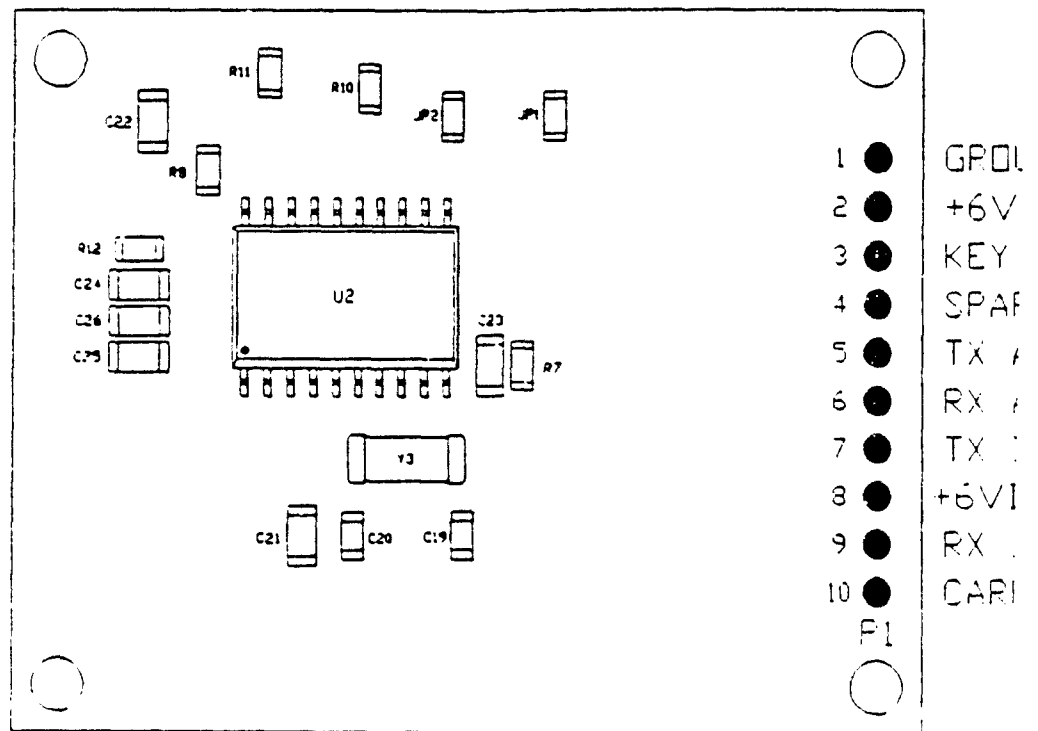
EMENT - FRONT ONLY)



DOCUMENT SUPPLIED BY MAGNAVOX

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC RADIO COMMUNICATIONS LINK PRINTED WIRING MASTER	
DRAWN GG	10-19-89		
CHECKED			
		SIZE B	FSCM NO.
		DRAWING NO. PMC-RAD-PWM	
		SCALE N/A	SHEET 1 OF 2

COMPONENT PLACEMENT - R



27

1

2

3

4

5

6

7

8

9

10

F1

- | | | |
|----|---|----------------|
| 1 | ● | GROUND |
| 2 | ● | +6VDC |
| 3 | ● | KEY LINE |
| 4 | ● | SPARE |
| 5 | ● | TX AUDIO |
| 6 | ● | RX AUDIO |
| 7 | ● | TX DATA |
| 8 | ● | +6VDC |
| 9 | ● | RX DATA |
| 10 | ● | CARRIER DETECT |

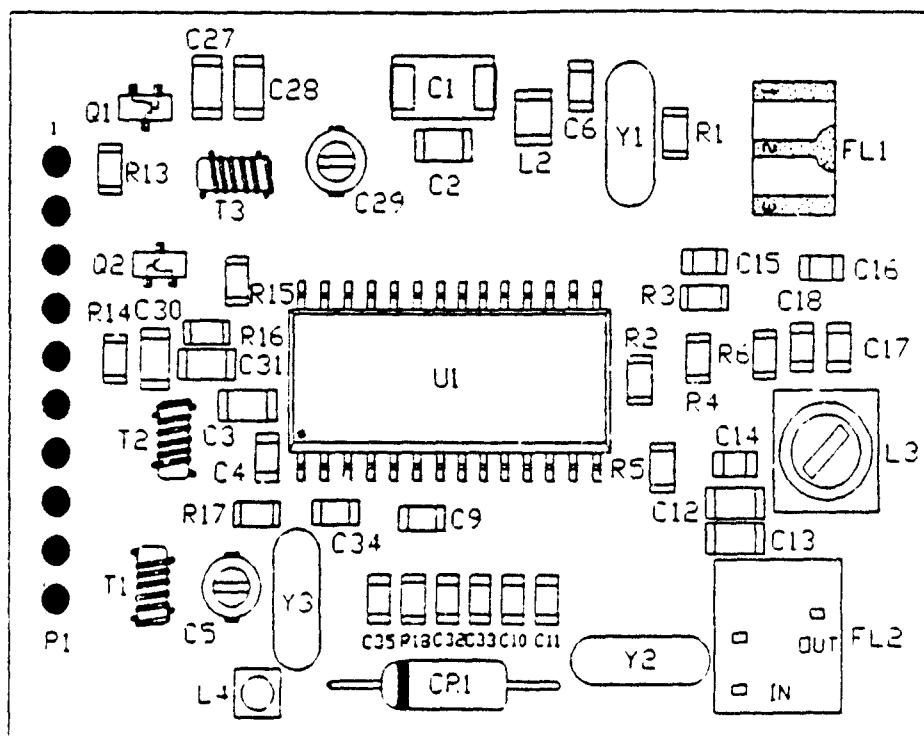
01

SIZE B	FSCM NO.	DRAWING NO. PMC-RAD-PWM
SCALE N/A		SHEET 2 OF 2

COMPONENT PLACEMENT - F

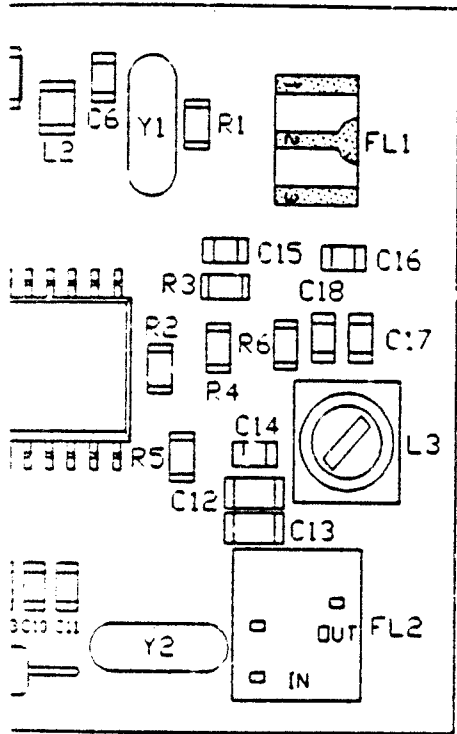
(-802 ONLY)

GROUND
 +5VDC
 KEY LINE
 CPARE
 TX AUDIO
 RX AUDIO
 TX DATA
 +5VDC
 RX DATA
 CARRIER DETECT



CONTRACT NO.	
DAMD17-87-C-719	
APPROVALS	
DRAWN	GG
CHECKED	

EMENT - FRONT
ONLY)



DOCUMENT SUPPLIED BY MAGNAVOX

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907		
APPROVALS	DATE	HHM RADIO COMMUNICATIONS LINK PRINTED WIRING MASTER		
DRAWN GG	10-19-88			
CHECKED				
		SIZE B	FSCM NO.	DRAWING NO. HHM-RAD-PWM
		SCALE N/A		SHEET 1 OF 2

SCHEMATIC DIAGRAMS
FOR THE
PERSONAL MONITOR AND COMMUNICATOR (PMC)

CONTRACT NO. DAMD17-87-C-7195

CDRL SEQUENCE NO. A009

9 October 1989

Prepared for:

Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick
Frederick, MD

Prepared by:

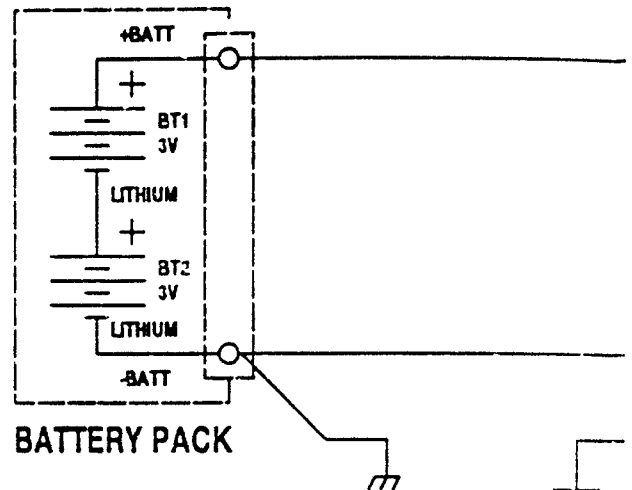
Institute for Interdisciplinary Engineering Studies
William A. Hillenbrand Biomedical Engineering Center
Purdue University
West Lafayette, In 47907

DATA W. A. HILLENBRAND BIOMED. DAMD17-87-C-7195 A009-DL 9/10/89
LIST ENGR. CENTER, PURDUE UNIV.

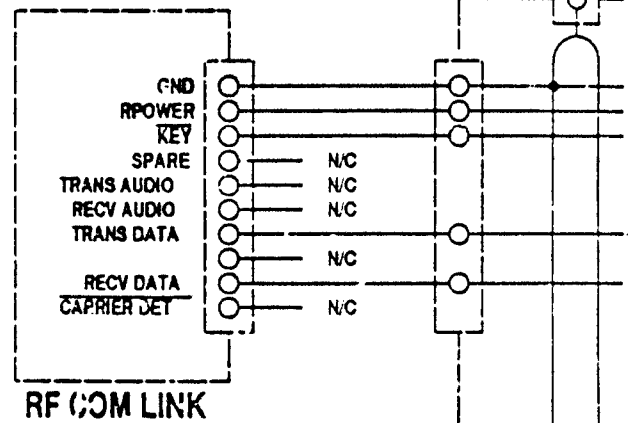
PMC/HHM ELECTRICAL ENGINEERING DRAWINGS LIST

SHEET 1 OF 1

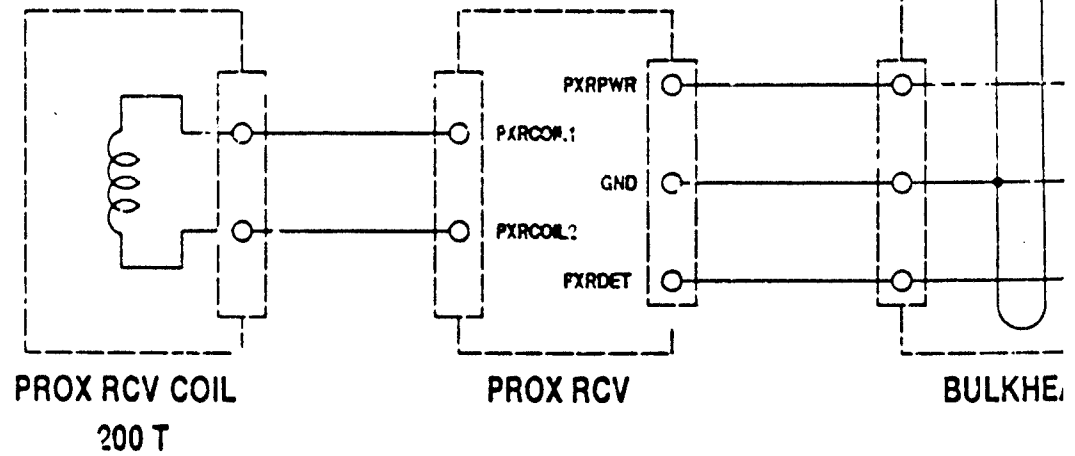
DWG SIZE	DOCUMENT NUMBER	SHEET NUMBER	DESCRIPTION
B	PMC-ICN-SCH	1	PMC INTERCONNECTION DIAGRAM
B	PMC-CPU-SCH	1-2	PMC MAIN BOARD SCHEMATIC DIAGRAM
B	PMC-CPU-PWM	1-8	PMC PRINTED WIRING MASTER
A	PMC-CPU-PL	1-2	PMC MAIN BOARD PARTS LIST
B	PMC-PXR-SCH	1	PMC PROXIMITY RECEIVER SCHEMATIC DIAGRAM
B	PMC-PXR-PWM	1-6	PMC PROX. RECEIVER PRINTED WIRING MASTER
A	PMC-PXR-PL	1	PMC PROXIMITY RECEIVER PARTS LIST
B	PMC-RAD-SCH	1-3	PMC RADIO LINK SCHEMATIC DIAGRAM
B	PMC-RAD-PWM	1-2	PMC RADIO LINK PRINTED WIRING MASTER
A	PMC-RAD-PL	1	PMC RADIO LINK PARTS LIST
B	PMC-BHD-PWM	1-6	PMC BULKHEAD BOARD PRINTED WIRING MASTER
B	HHM-ICN-SCH	1	HHM INTERCONNECTION DIAGRAM
B	HHM-CPU-SCH	1	HHM MAIN BOARD SCHEMATIC DIAGRAM
B	HHM-CPU-PWM	1-6	HHM PRINTED WIRING MASTER
A	HHM-CPU-PL	1	HHM MAIN BOARD SCHEMATIC PARTS LIST
B	HHM-RAD-SCH	1-3	HHM RADIO LINK SCHEMATIC DIAGRAM
B	HHM-RAD-PWM	1-2	HHM RADIO LINK PRINTED WIRING MASTER
A	HHM-RAD-PL	1	HHM RADIO LINK PARTS LIST
B	HHM-ANT-SCH	1	HHM RADIO ANTENNA SCHEMATIC DIAGRAM



BATTERY PACK



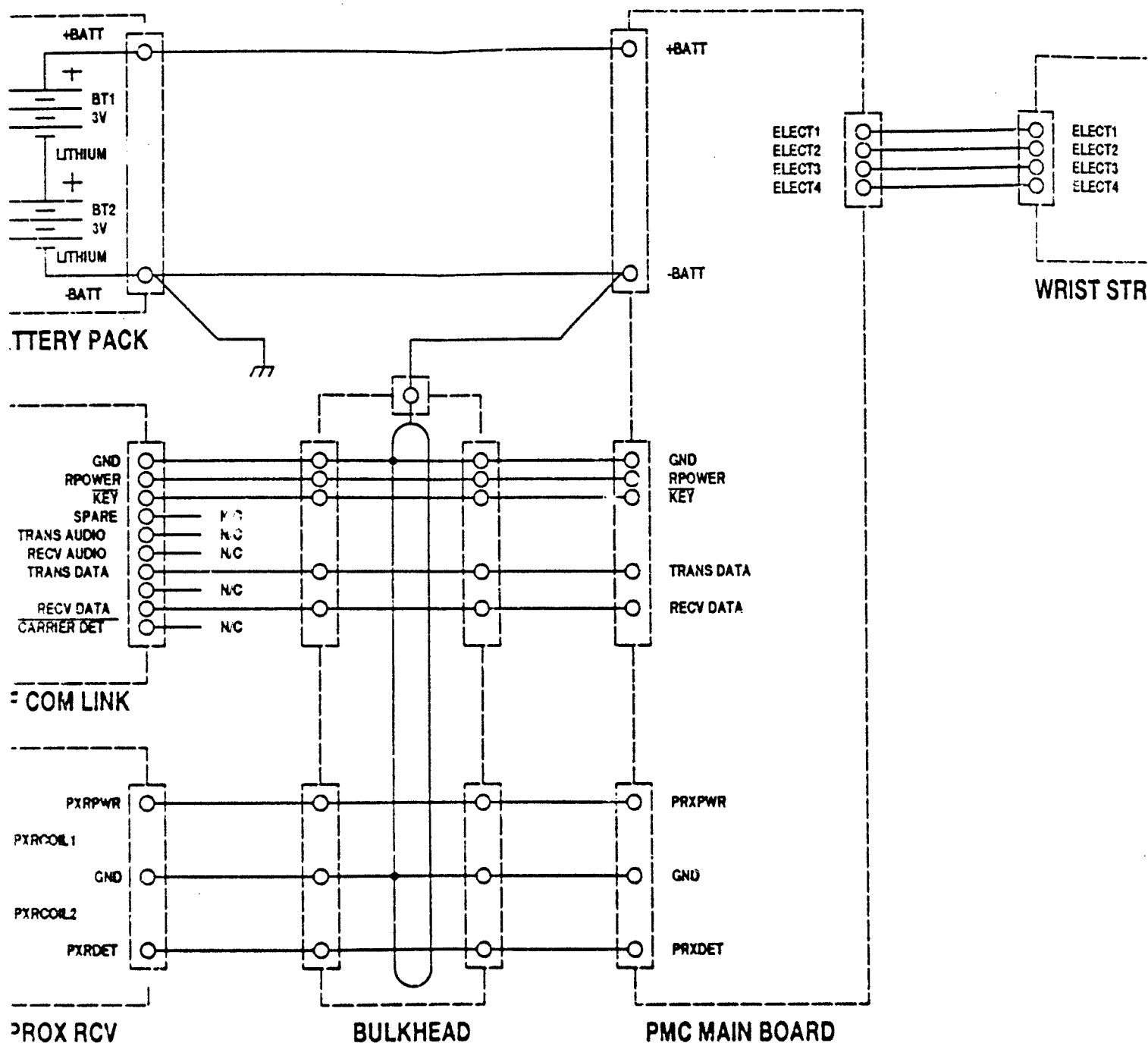
RF COM LINK



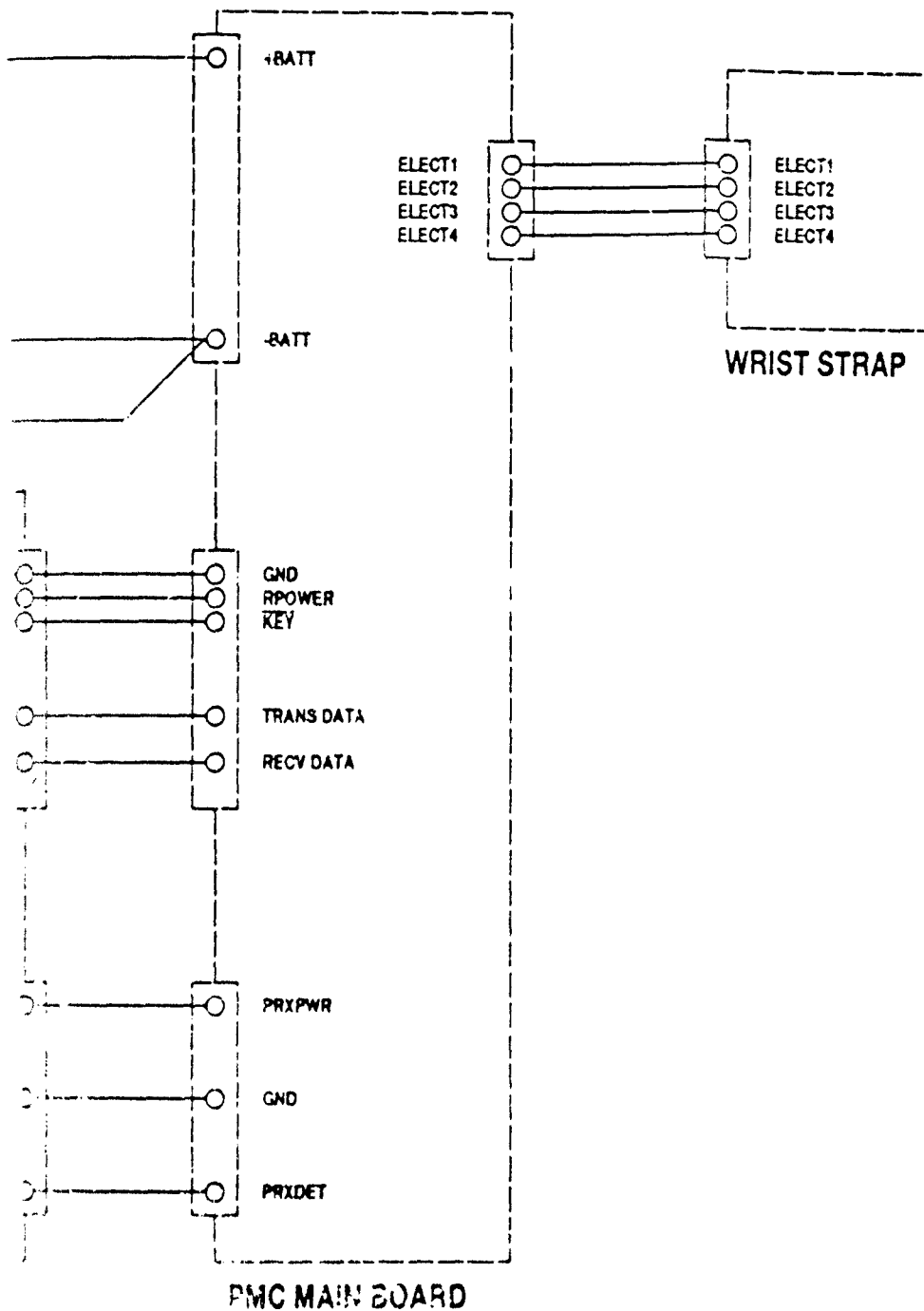
PROX RCV COIL
200 T

PROX RCV

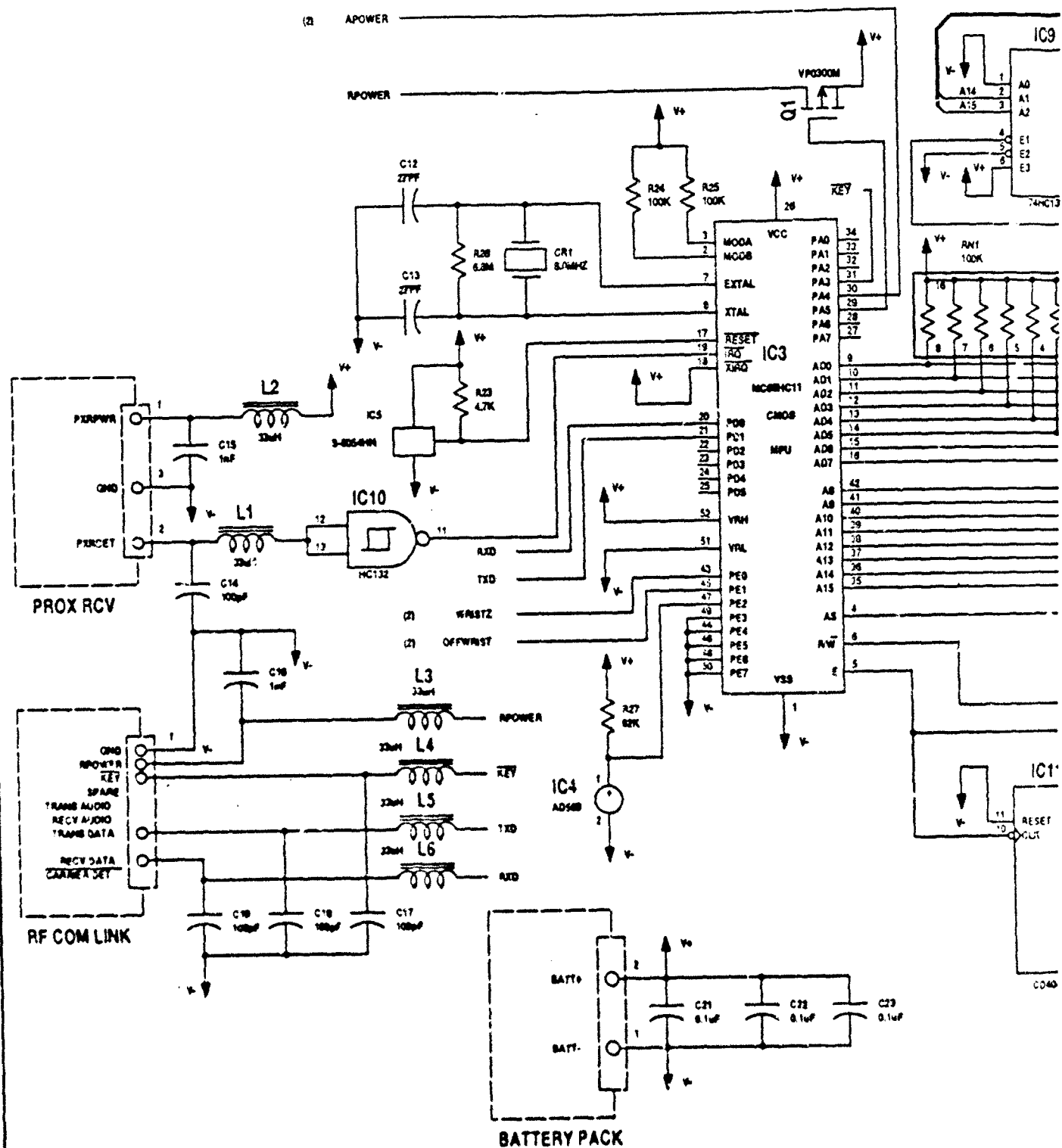
BULKHEAD

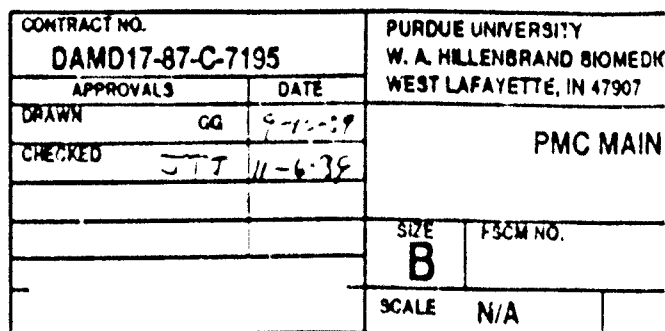


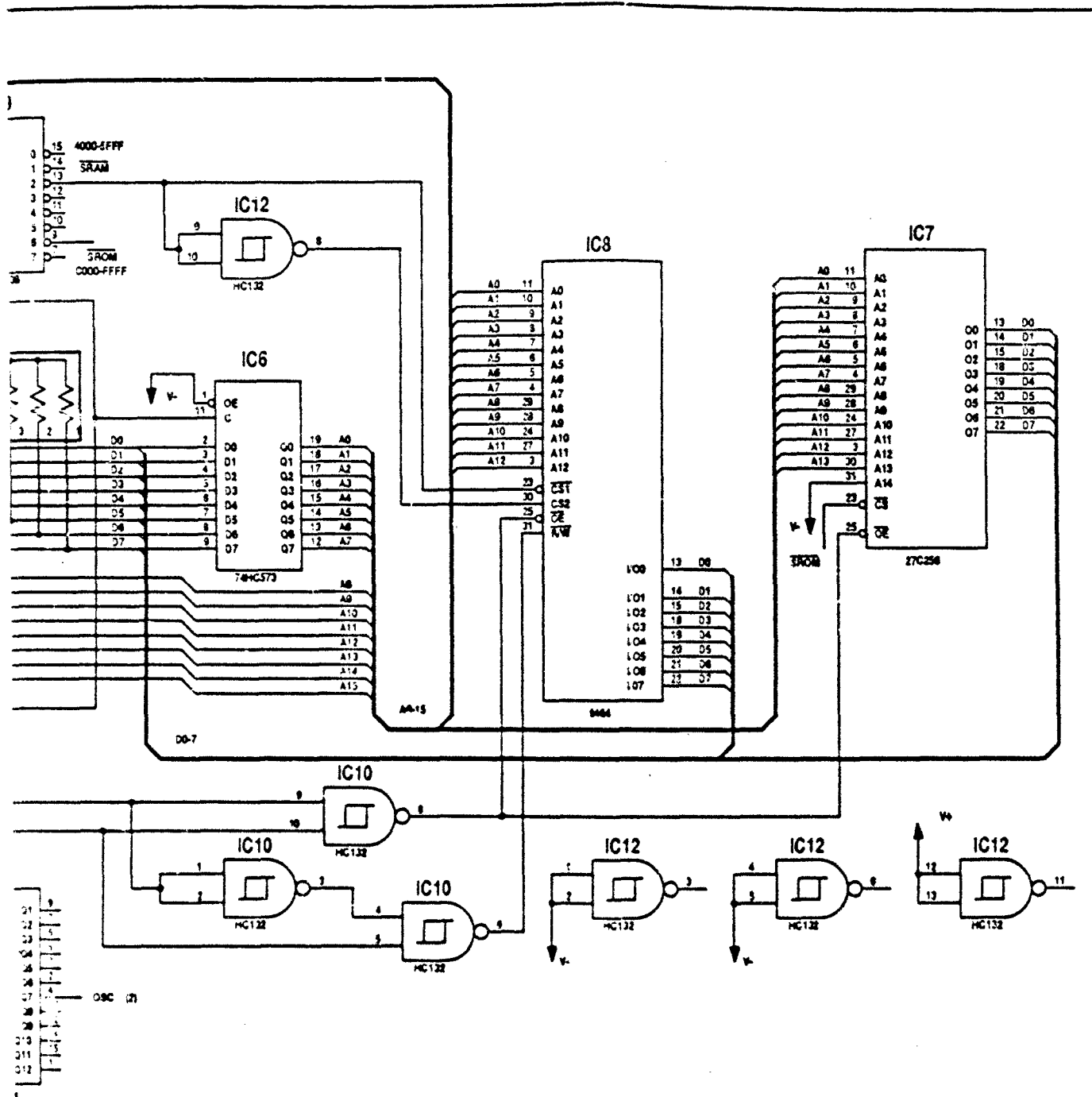
CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDI WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC INTERCC	
DRAWN <i>GG</i>	<i>7-10-89</i>		
CHECKED <i>ITS</i>	<i>4-6-89</i>		
		SIZE B	FSCM NO.
		SCALE	N/A



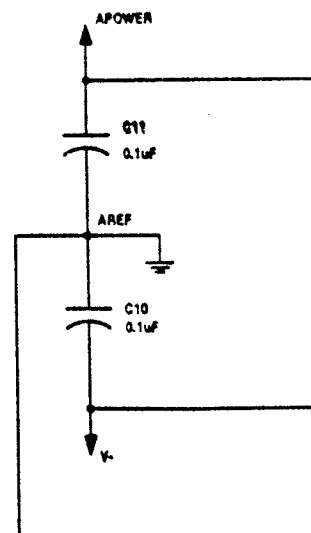
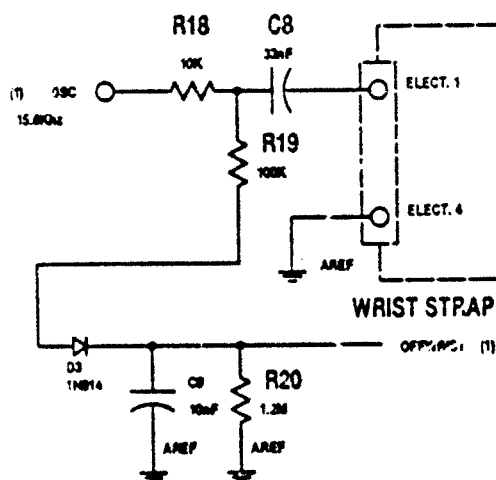
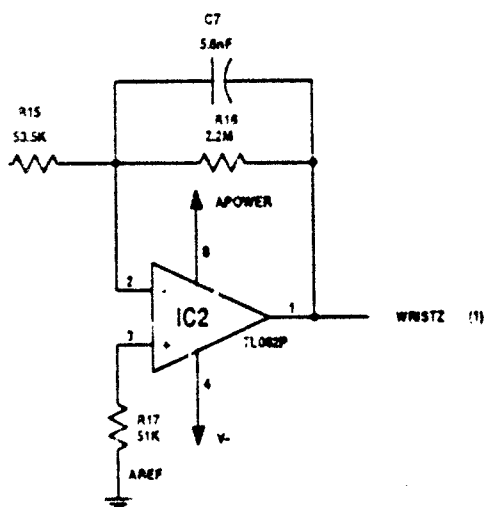
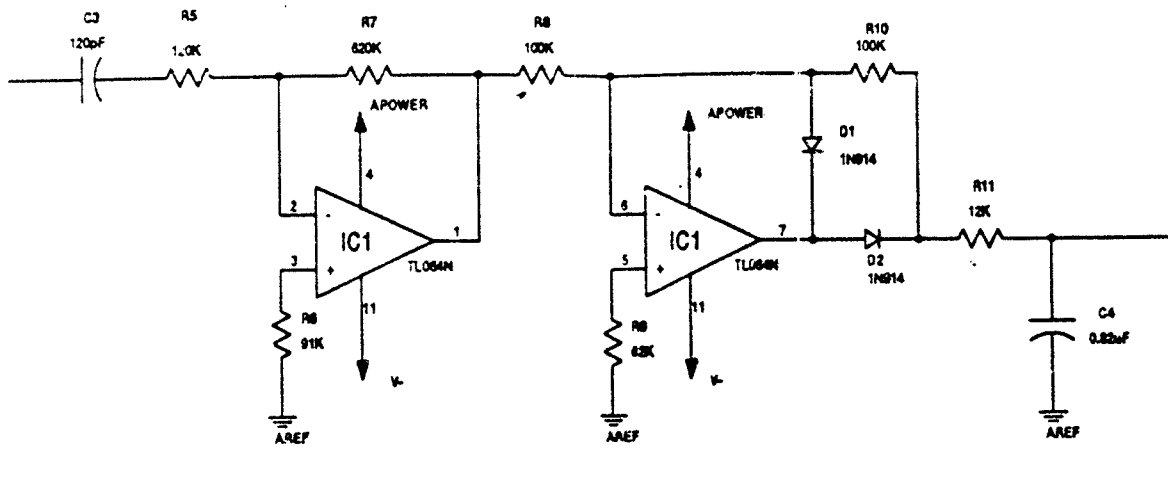
PROJECT NO. DAHD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC INTERCONNECTION DIAGRAM	
DRAWN GG	4-10-89		
CHECKED JTS	4-10-89		
		SIZE B	DRAWING NO. PMC-ICN-SCH
		SCALE N/A	SHEET 1 OF 1



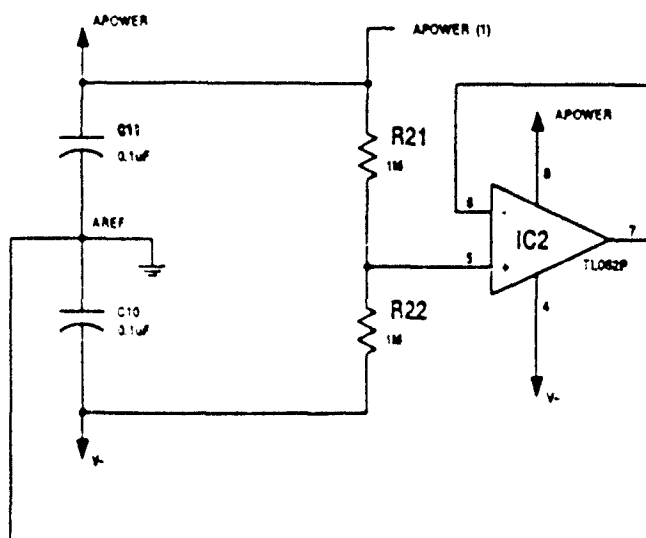
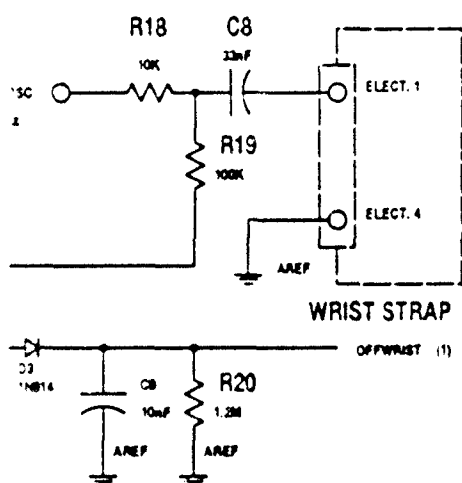
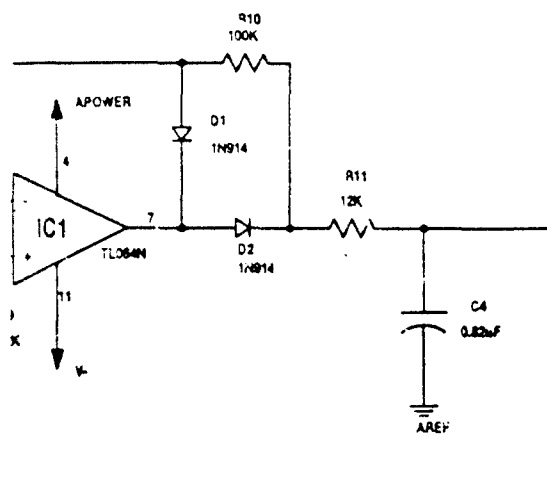




CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC MAIN BOARD SCHEMATIC	
DRAWN OG	9-10-89		
CHECKED JTT	11-6-89	PMC-CPU-SCH	
SIZE B	FSCM NO.	DRAWING NO.	
SCALE N/A		SHEET 1 OF 2	

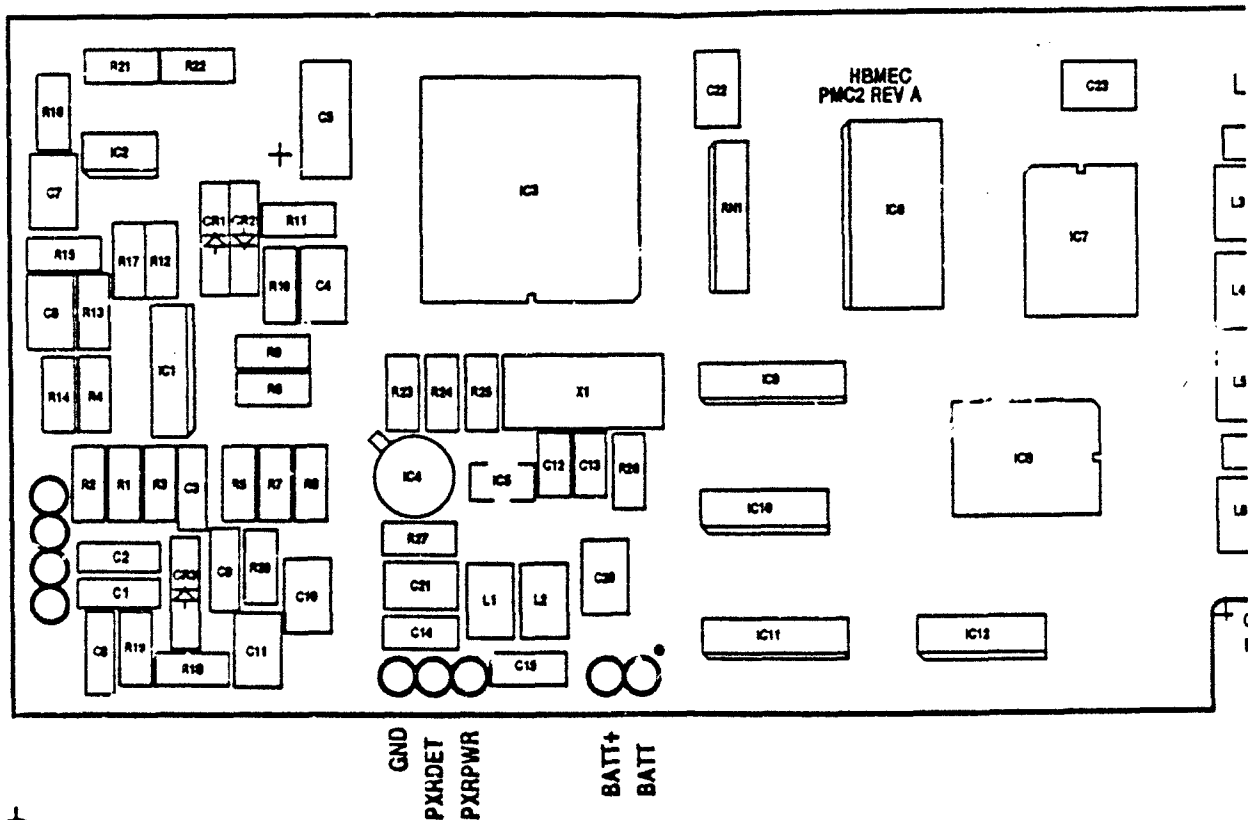


SIZE B	FSCM NO.
SCALE	N/A



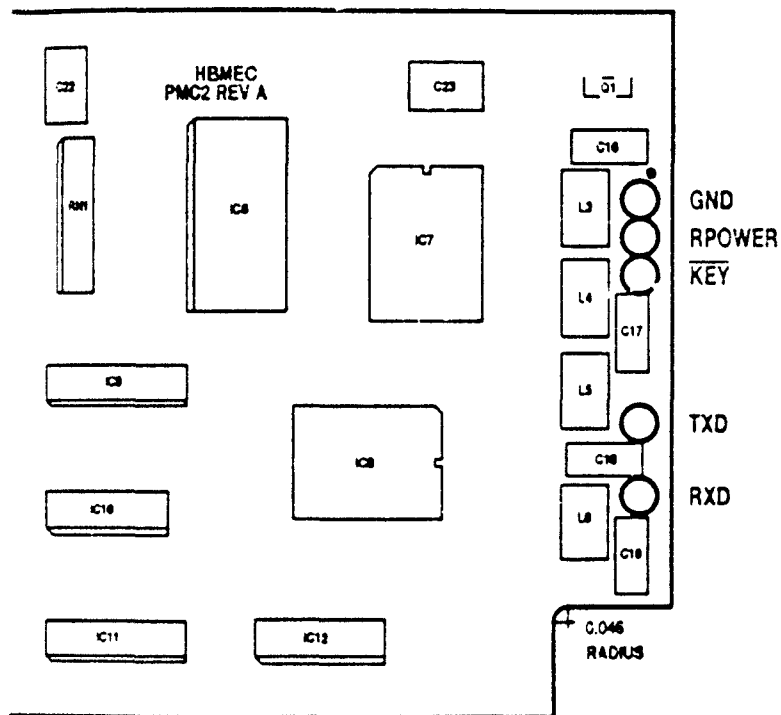
S/E B	FSCM NO.	DRAWING NO. PMC-CPU-SCH
SCALE N/A		SHEET 2 OF 2

ELECT4
ELECT3
ELECT2
ELECT1



REDUCE TO 3.750 INCHES

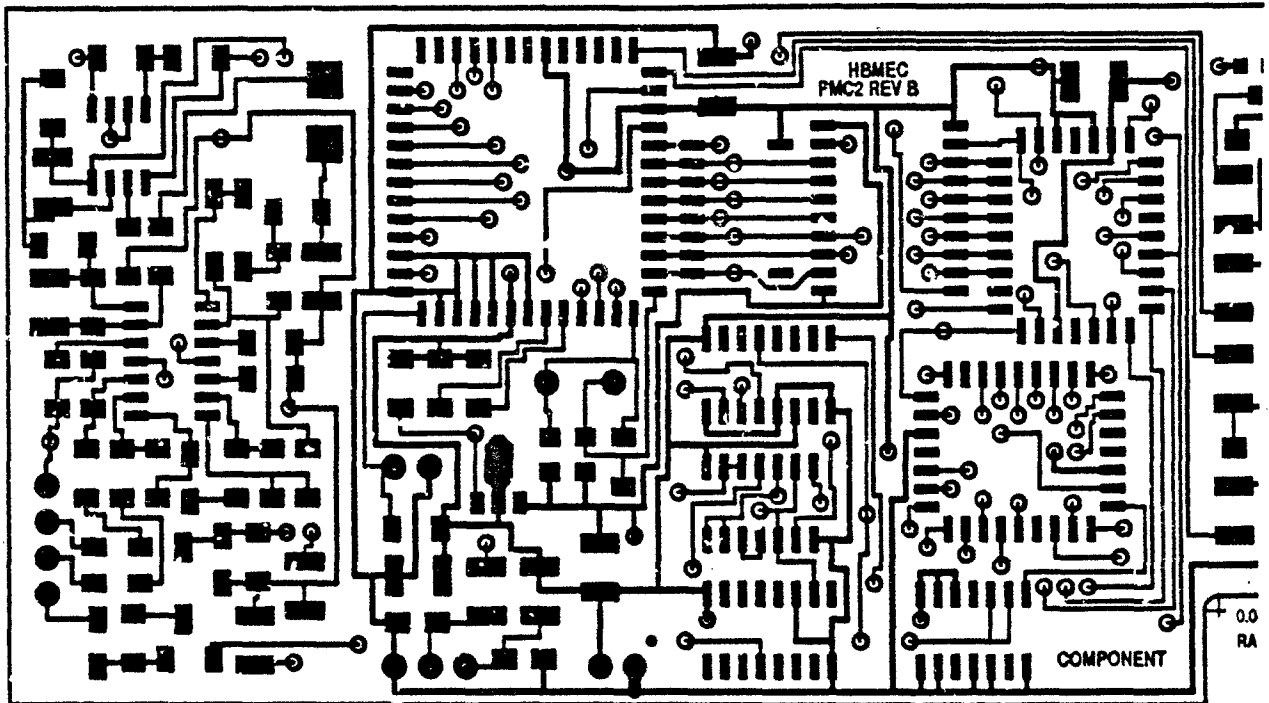
CONTRACT NO.		
DAMD17-87-C-7195		
APPROVALS		
DRAWN	GG	9-
CHECKED	JTJ	11-



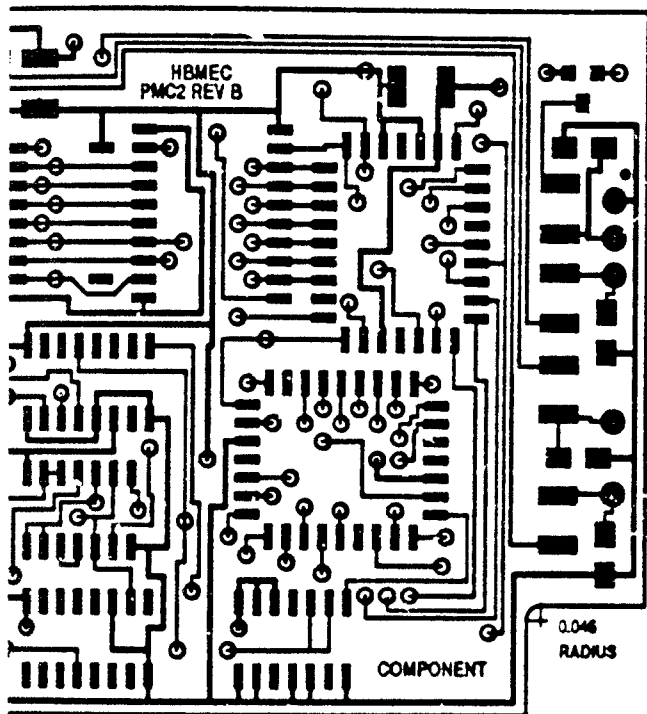
3.750 INCHES

PARTS PLACEMENT

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC MAIN BOARD PRINTED WIRING MASTER	
DRAWN GG	9-10-89		
CHECKED JIT	11-6-89		
		SIZE B	FSCM NO.
		DRAWING NO. PMC-CPU-PWM	
		SCALE 2/1	SHEET 1 OF 8



REDUCE TO 3.750 INCHES

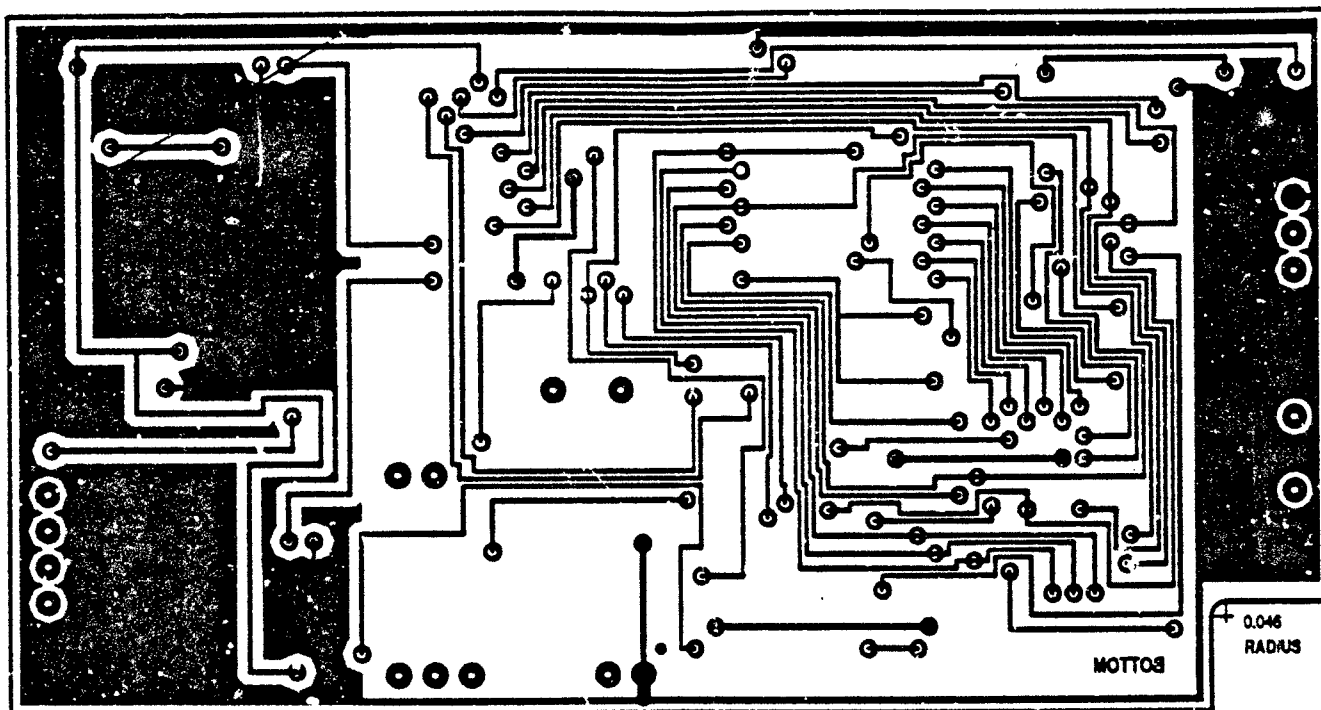


1.50 INCHES

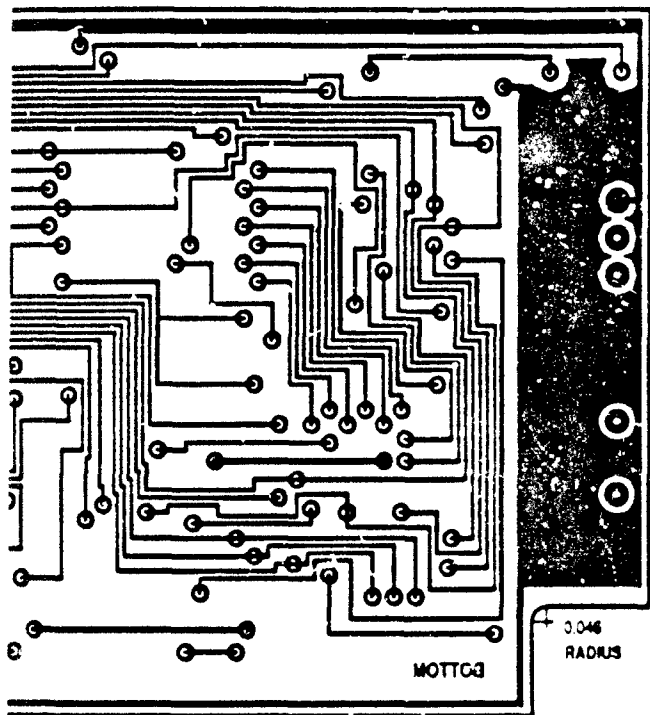


TOP (COMPONENT) LEVEL

SIZE B	FSCM NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2/1		SHEET 2 OF 8



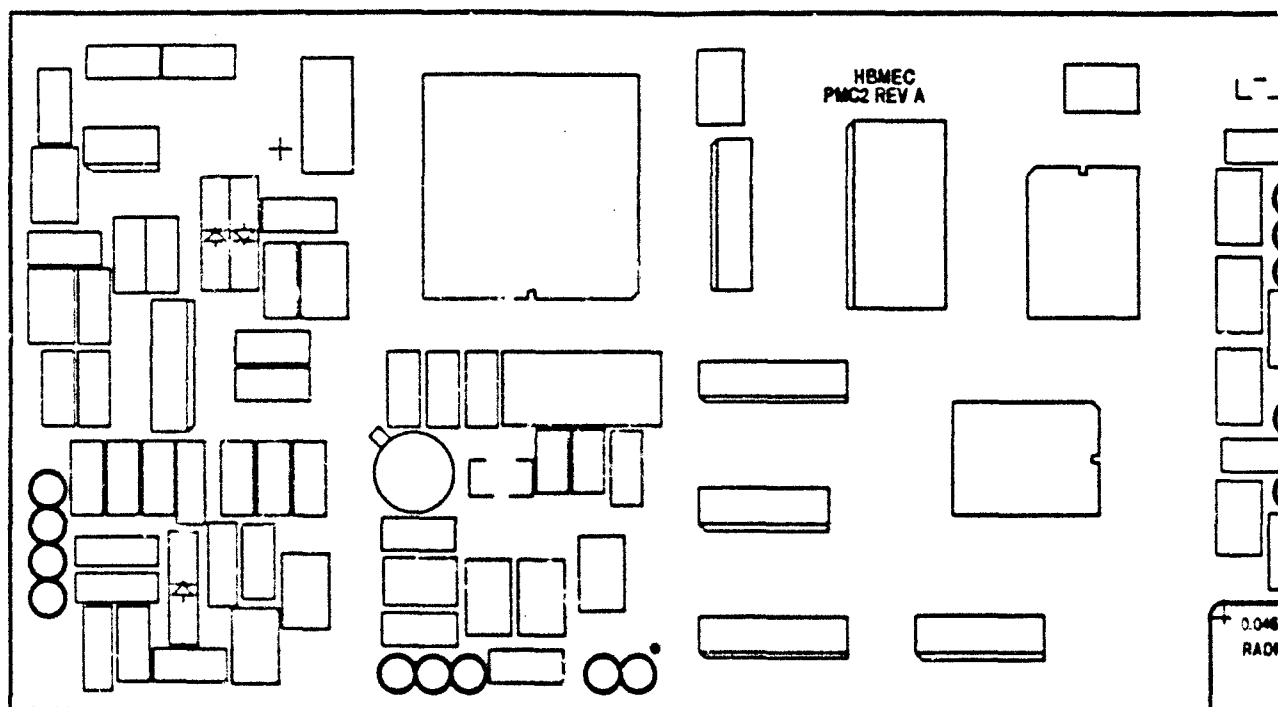
REDUCE TO 1.750 INCHES



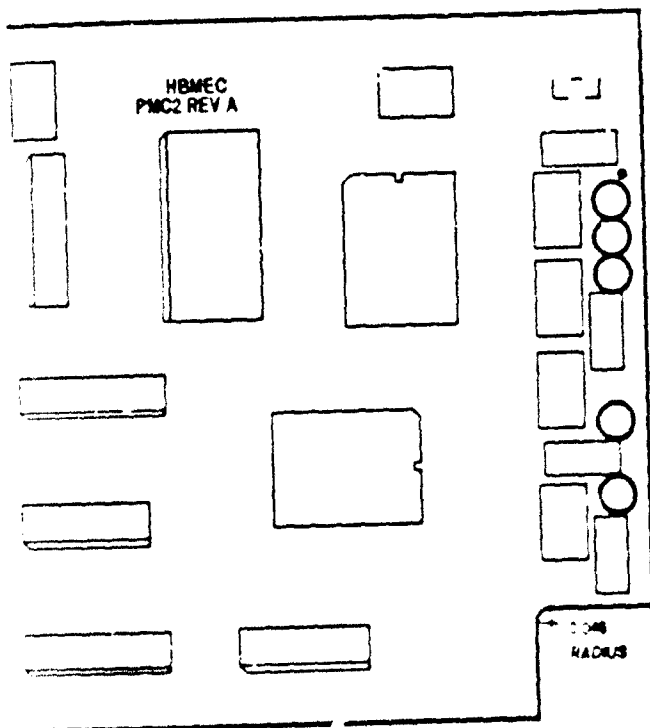
.750 INCHES

BOTTOM (SOLDER) LAYER

SIZE B	PSCM NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2/1		SHEET 3 OF 8



REDUCE TO 3.750 INCHES

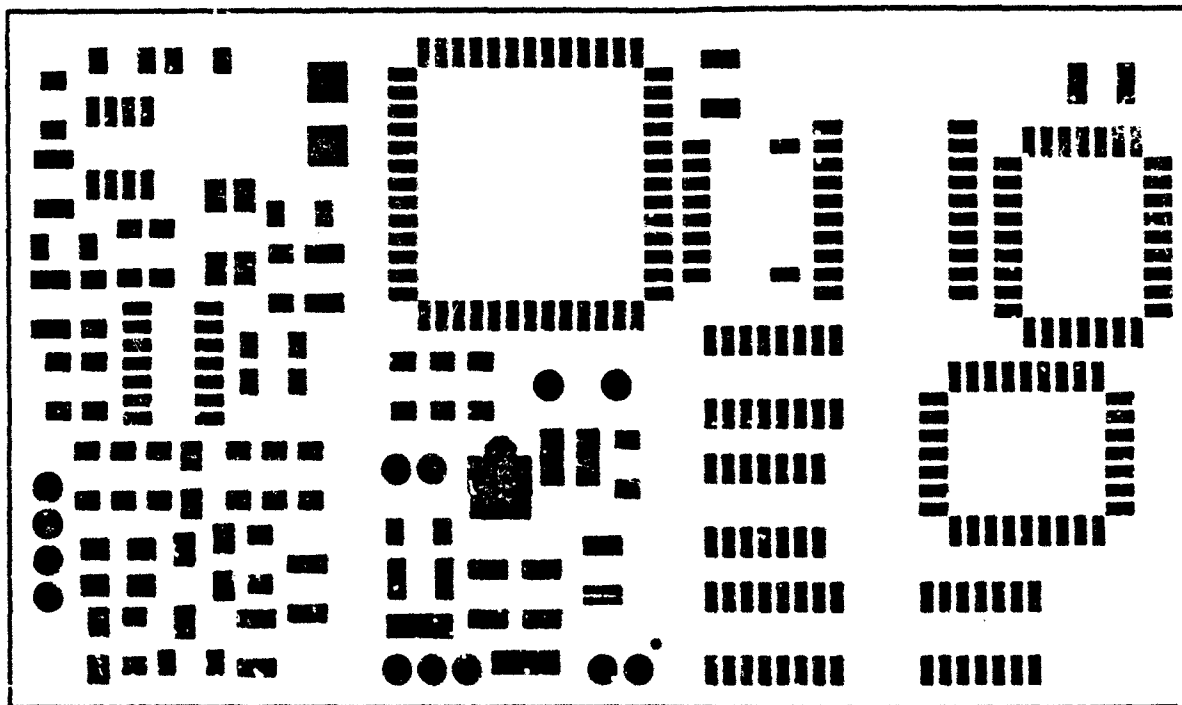


0.100 INCHES

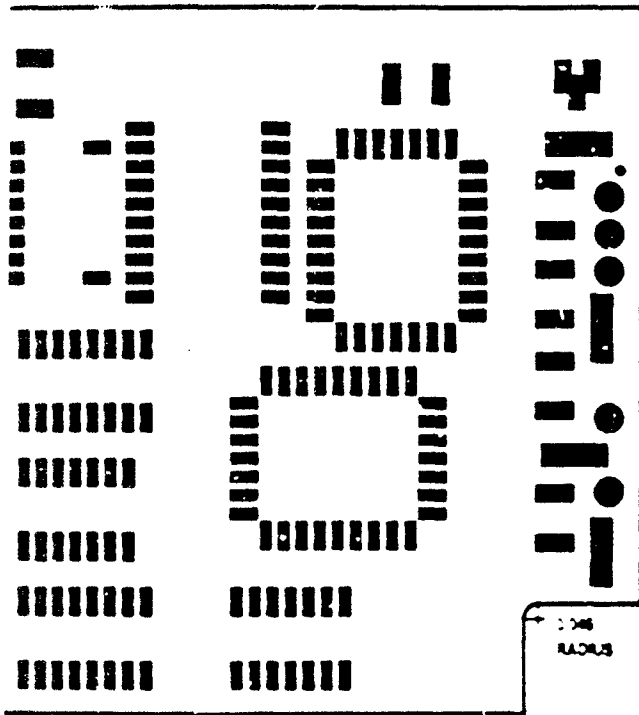


SILK SCREEN LAYER

REV B	FIGURE NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2:1		SHEET 4 OF 8



REDUCE TO 1.750 INCHES

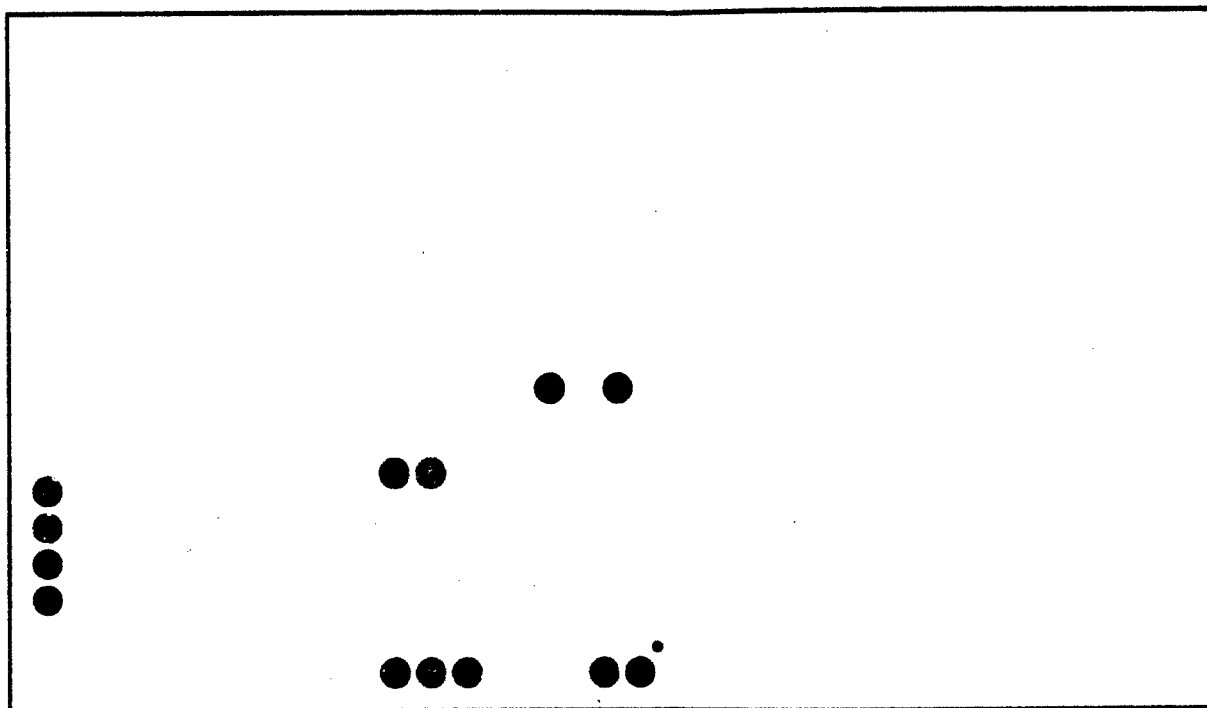


100 INCHES

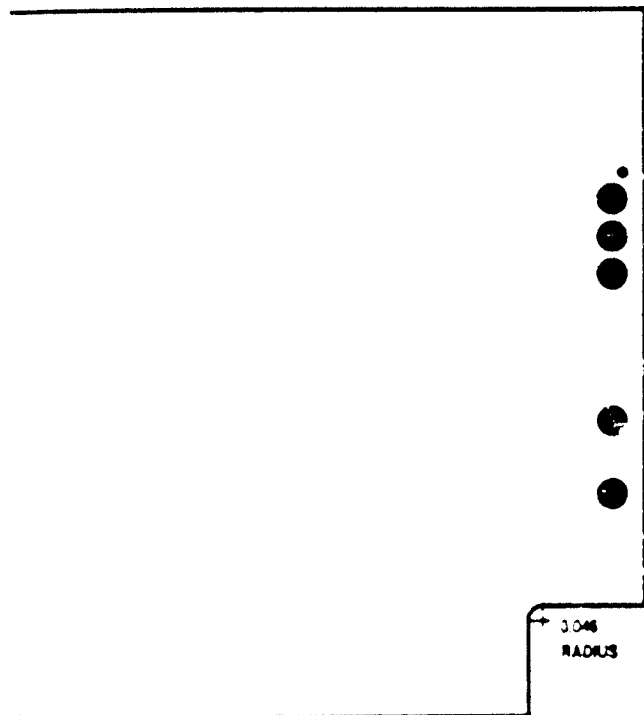


SOLDER MASK LAYER (TOP)

SET B	FIGURE NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2:1		SHEET 5 OF 8



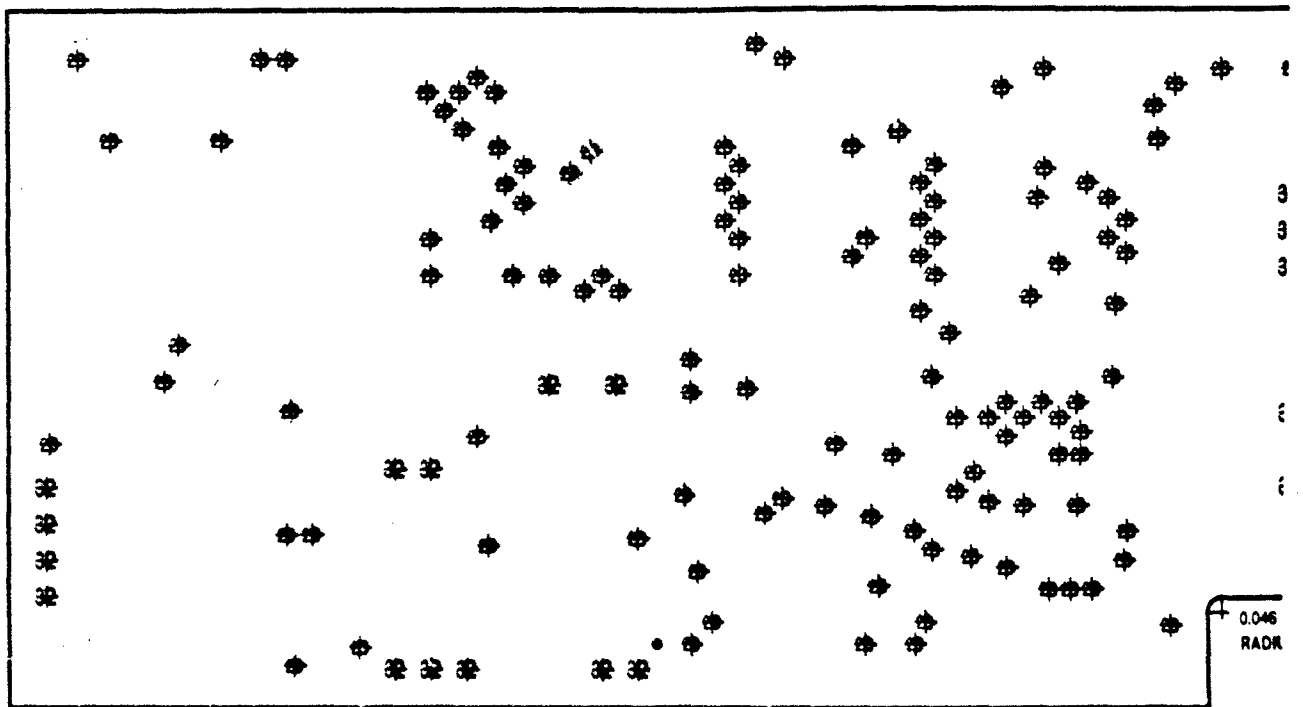
REDUCE TO 1.750 INCHES



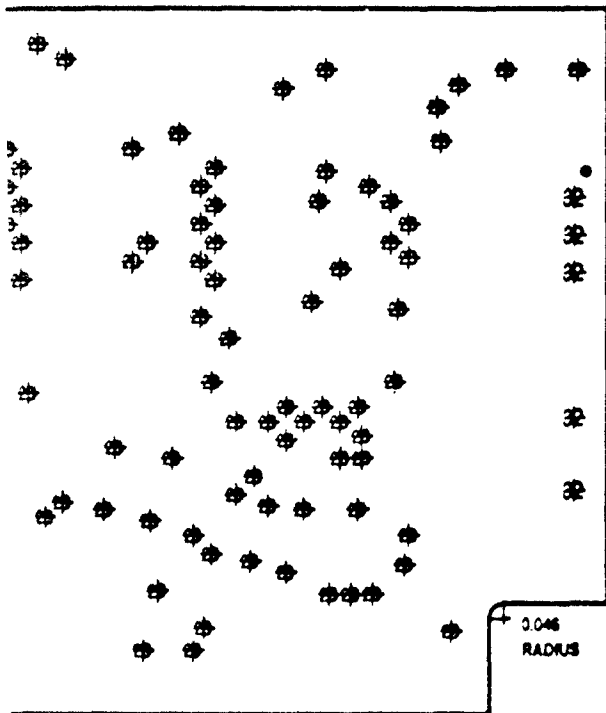
750 INCHES

SOLDER MASK LAYER (BOTTOM)

SITE B	PLUM NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2/1		SHEET 6 OF 8



REDUCE TO 3.750 INCHES

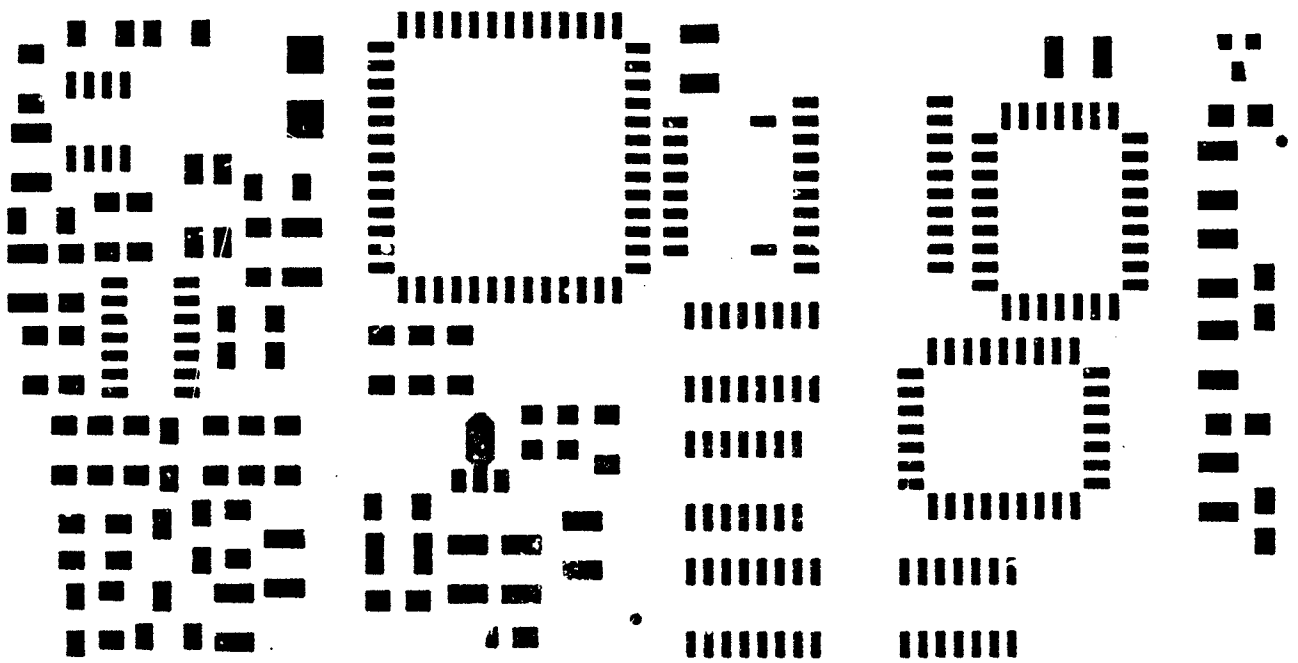


NCHES

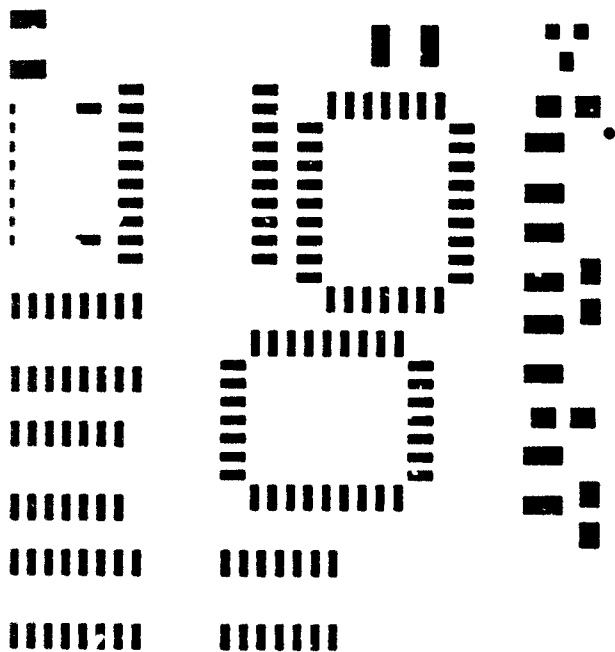


DRILL MASTER LAYER

SIZE B	FSCM NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2/1		SHEET 7 OF 8



REDUCE TO 3.750 INCHES



0 INCHES



SOLDER PASTE LAYER

SIZE B	FSCM NO.	DRAWING NO. PMC-CPU-PWM
SCALE 2/1		SHEET 8 OF 8

PARTS W. A. HILLENBRAND BIOMED. DAMD17-87-C-7195 PMC-CPU-PL 9/10/89
 LIST ENGR. CENTER, PURDUE UNIV.

PMC MAIN BOARD

SHEET 1 OF 2

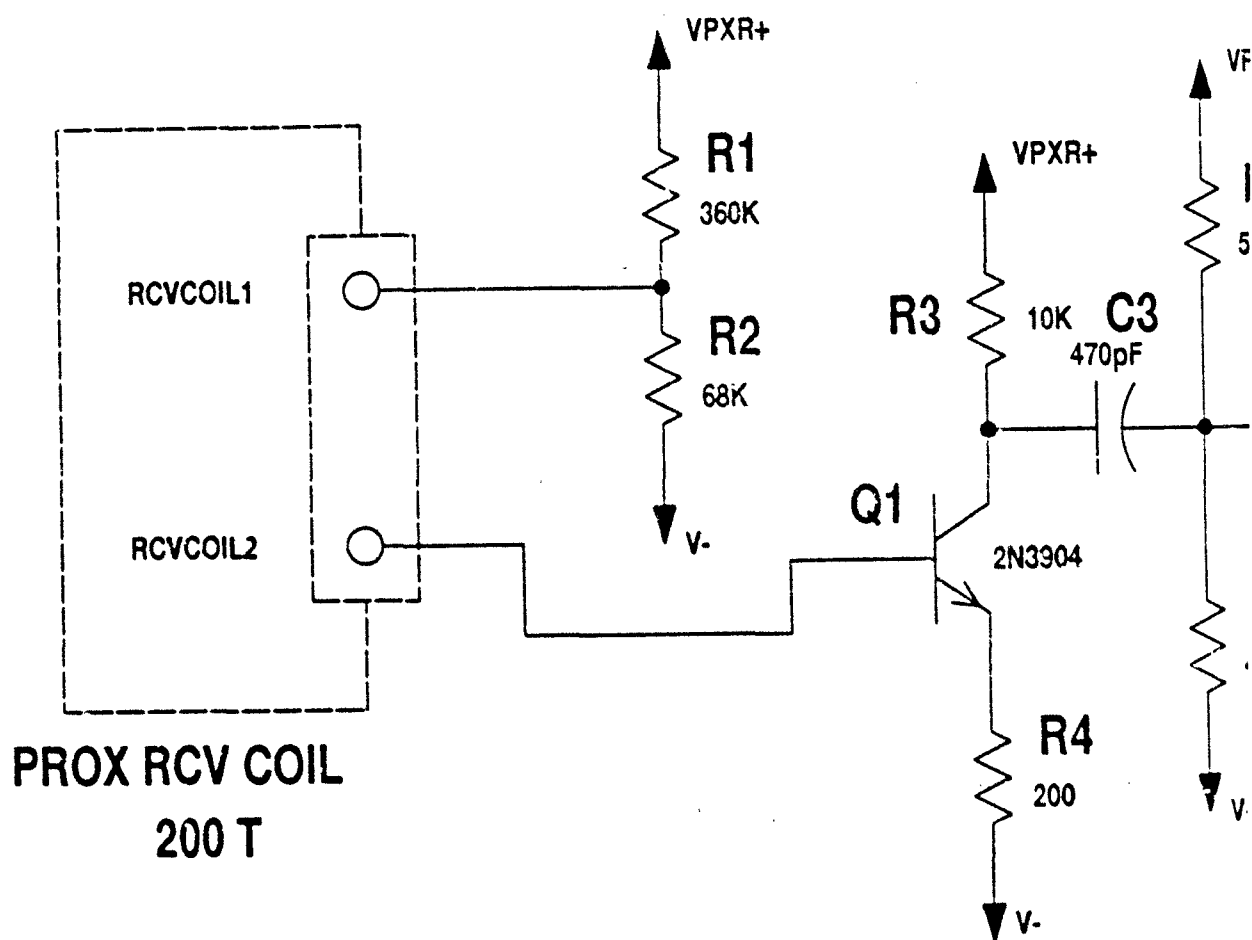
QTY	COMP-NAME	REFERENCE-DESIGNATOR	DESCRIPTION
1	68HC11A1FN	IC3	
8	HC132	IC12 IC10 IC10 IC10 IC12 IC12 IC12 IC10	
11	RESV	R24 R25 RN1H RN1F RN1G RN1E RN1A RN1D RN1C RN1B R19	VAL=100K
1	RESV	R23	VAL=4.7K
1	RESV	R26	VAL=6.8M
2	RESV	R27 R9	VAL=62K
1	RESV	R20	VAL=1.2M
2	RESV	R17 R14	VAL=51K
1	RESV	R4	VAL=470K
1	RESV	R6	VAL=91K
2	RESV	R21 R22	VAL=1M
1	74HC573	IC6	
1	74HC138	IC9	
1	27C256P	IC7	
5	CAPV	C3 C23 C22 C21 C24	VAL=0.1uF
4	CAPV	C14 C18 C19 C17	VAL=100pF
2	CAPV	C15 C16	VAL=1nF
1	XTALV	CR1	VAL=8.0MHZ
1	AD589	IC4	
2	CAPH	C12 C13	VALUE=27PF
1	CAPH	C8	VALUE=33nF

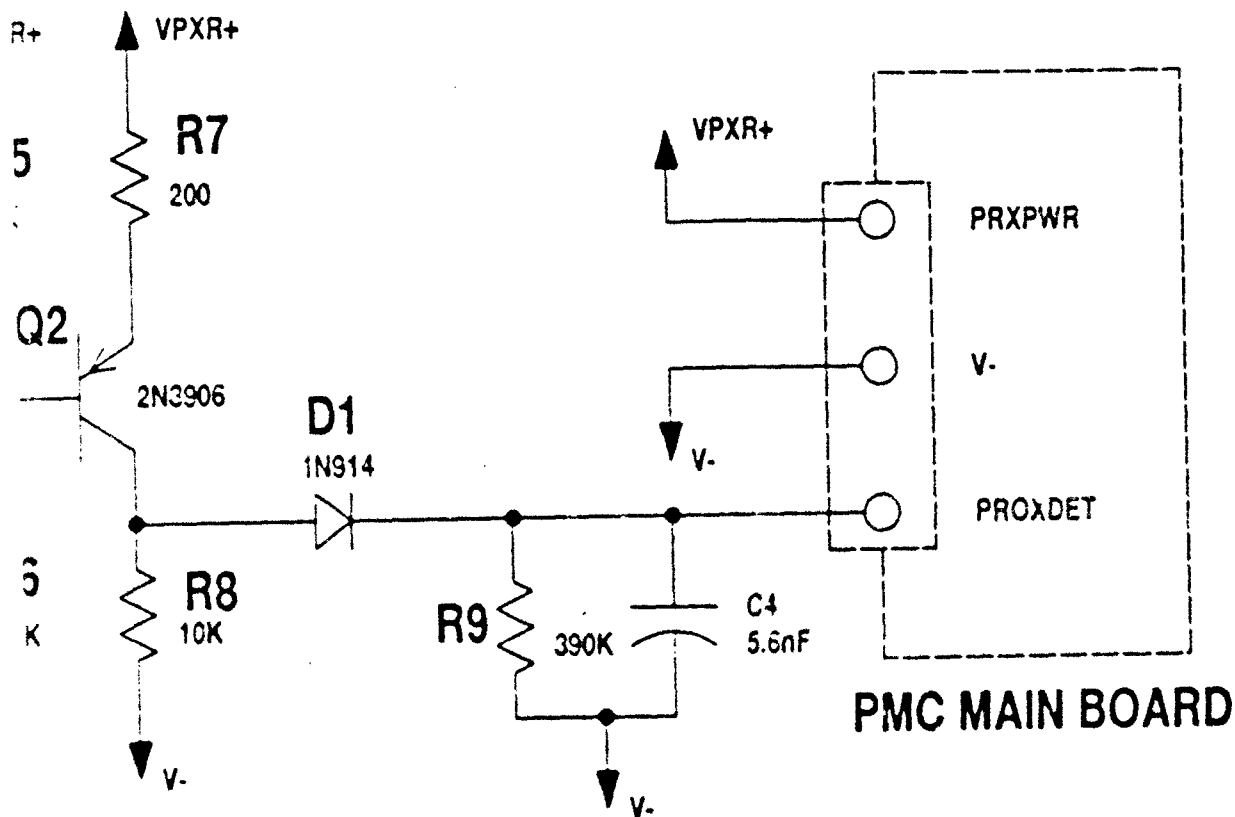
PARTS W. A. HILLENBRAND BIOMED. DAMD17-87-C-7195 PMC-CPU-PL 9/10/89
 LIST ENGR. CENTER, PURDUE UNIV.

PMC MAIN BOARD

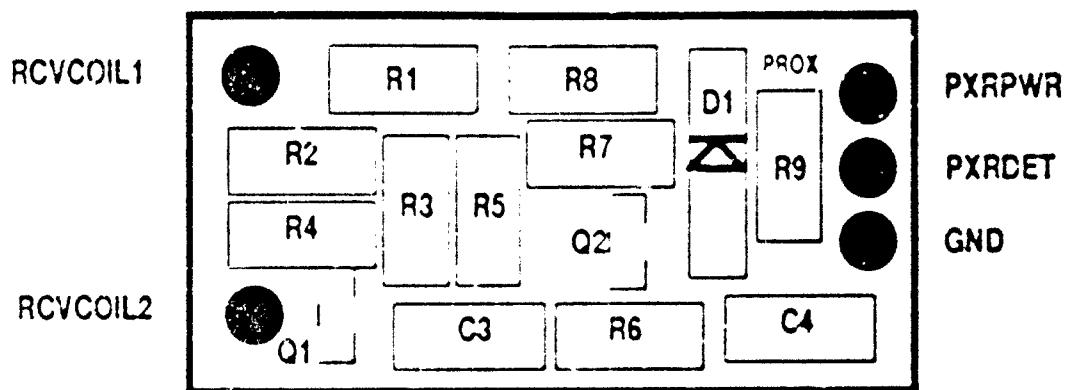
SHEET 2 OF 2

QTY	COMP-NAME	REFERENCE-DESIGNATOR	DESCRIPTION
---	-----	-----	-----
2	CAPH	C7 C6	VALUE=5.6nF
2	CAPH	C1 C2	VALUE=820pF
1	CAPH	C3	VALUE=120pF
1	PMOSSGD	Q1	
6	CHOKE	L1 L3 L4 L5 L6 L2	VAL=33uH
1	8464P	IC8	
1	C4040B	IC11	
1	S8054HN	IC5	
1	RESH	R18	VAL=10K
2	RESH	R16 R13	VAL=2.2M
2	RESH	R15 R12	VAL=53.5K
2	RESH	R2 R1	VAL=47K
1	RESH	R3	VAL=470K
1	RESH	R5	VAL=120K
1	RESH	R7	VAL=620K
2	RESH	R8 R10	VAL=100K
1	RESH	R01	VAL=12K
2	CAPV	C11 C10	VAL=0.1uF
1	CAPV	C9	VAL=10nF
1	CAPV	C4	VAL=0.82uF
3	DIODE	D3 D2 D1	
4	TL064N	IC1 IC1 IC1 IC1	
1	POLCAP	C5	VAL=2.7uf
2	TL062P	IC2 IC2	





CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC PROXIMITY RECEIVER SCHEMATIC	
DRAWN <i>GG</i>	<i>4-10-88</i>		
CHECKED <i>JTT</i>	<i>11-6-88</i>		
SIZE B		PSOM NO.	DRAWING NO. PMC-PXR-SCH
SCALE N/A		SHEET 1 OF 1	

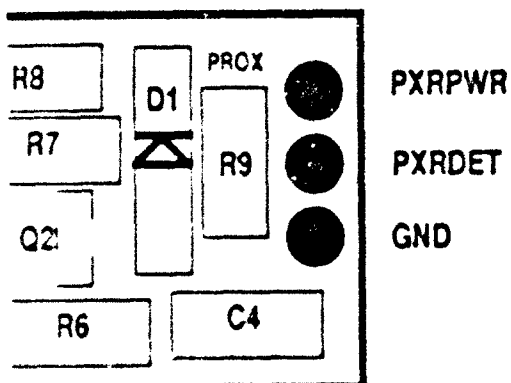


REDUCE TO 1.0"



PARTS PLACED

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. MILLENBRAND BIOME WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC PRC PRINTED	
DRAWN	DATE		
CHECKED	DATE	SITE B	
		SCALE 4/1	

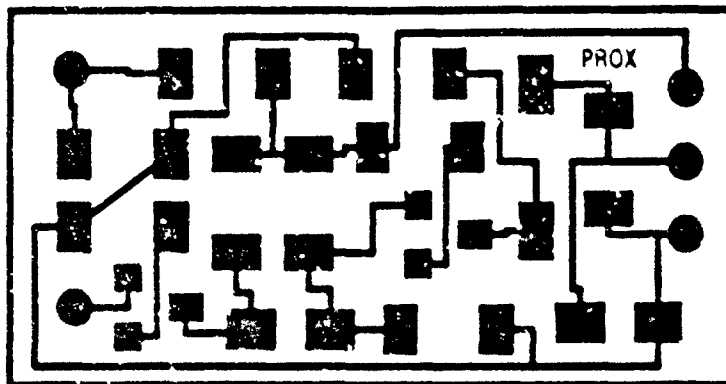


0 1.0"



PARTS PLACEMENT

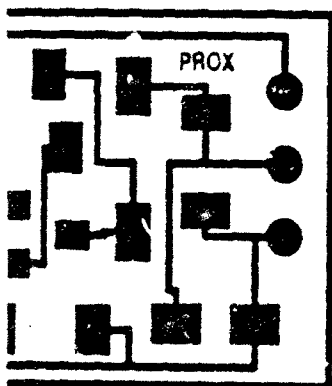
CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC PROXIMITY RECIEVER PRINTED WIRING MASTER	
DRAWN GG	2-20-85		
CHECKED JTT	11-1-85	SIZE B PSCM NO. DRAWING NO. PMC-PXR-PWM	
		SCALE 4:1	SHEET 1 OF 6



REDUCE TO 1.0"



SHEET 8	PROCESS NO.
SCALE	4/1

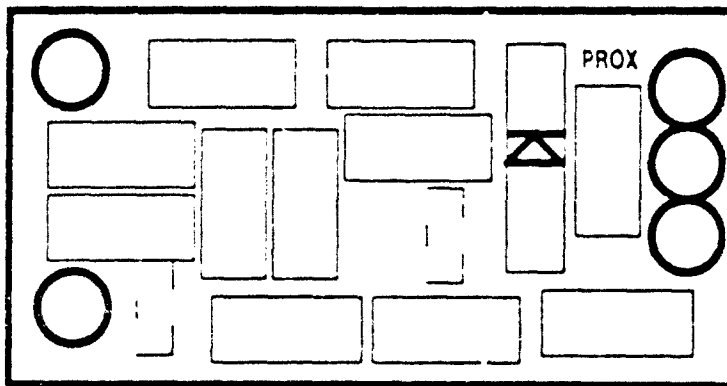


.0"



TOP (COMPONENT) LEVEL

SIZE B	FSCM NO.	DRAWING NO. PMC-PXR-PWM
SCALE 4/1		SHEET 2 OF 6

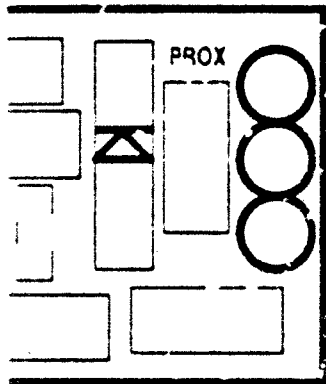


REDUCE TO 1.0"



SIL

SIZE	FSCM NO.	
B		
SCALE	4/1	

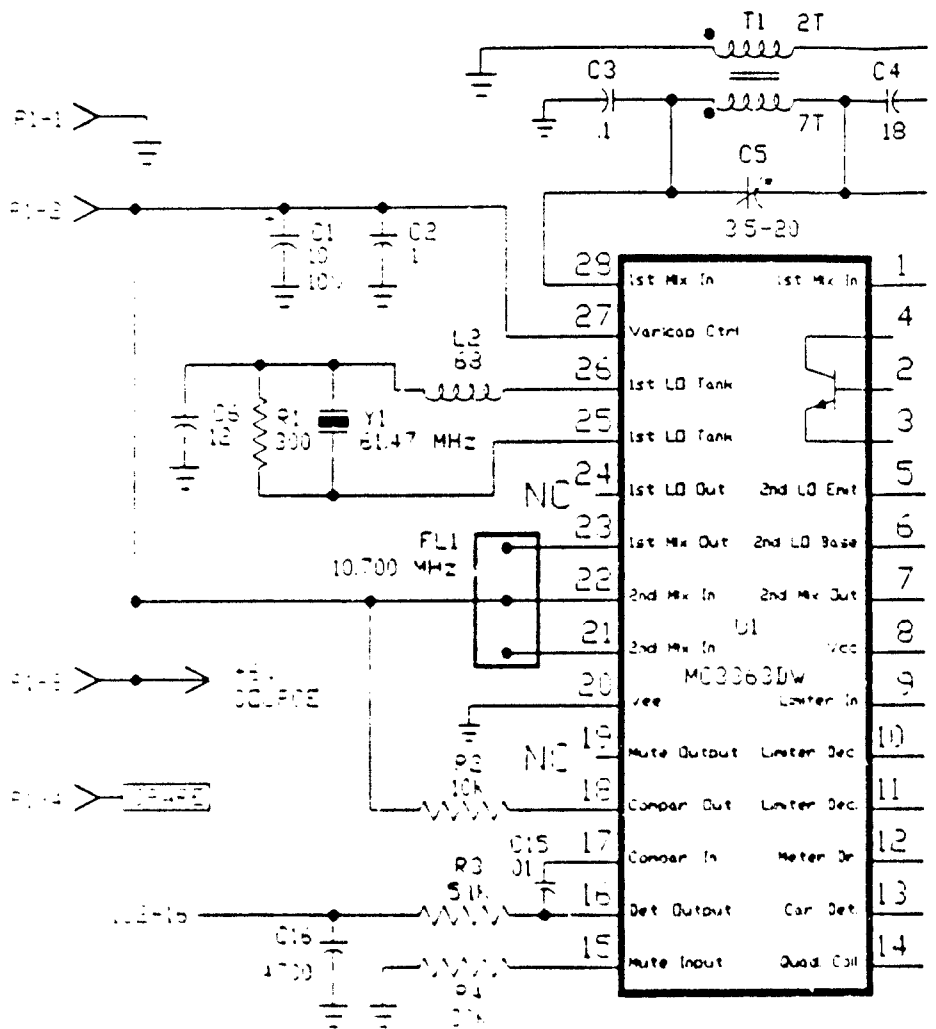


0"

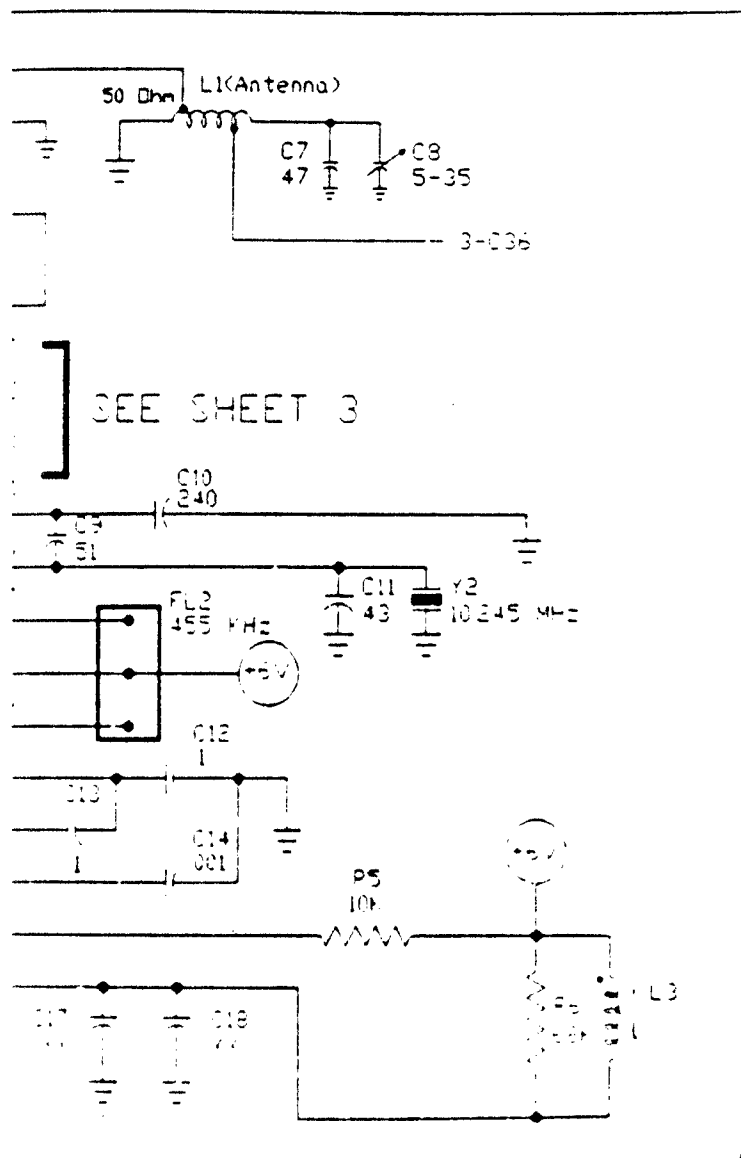


SILK SCREEN LAYER

SIZE B	PSCM NO.	DRAWING NO. PMC-PXR-PWM
SCALE 4/1		SHEET 3 OF 6



14-00000

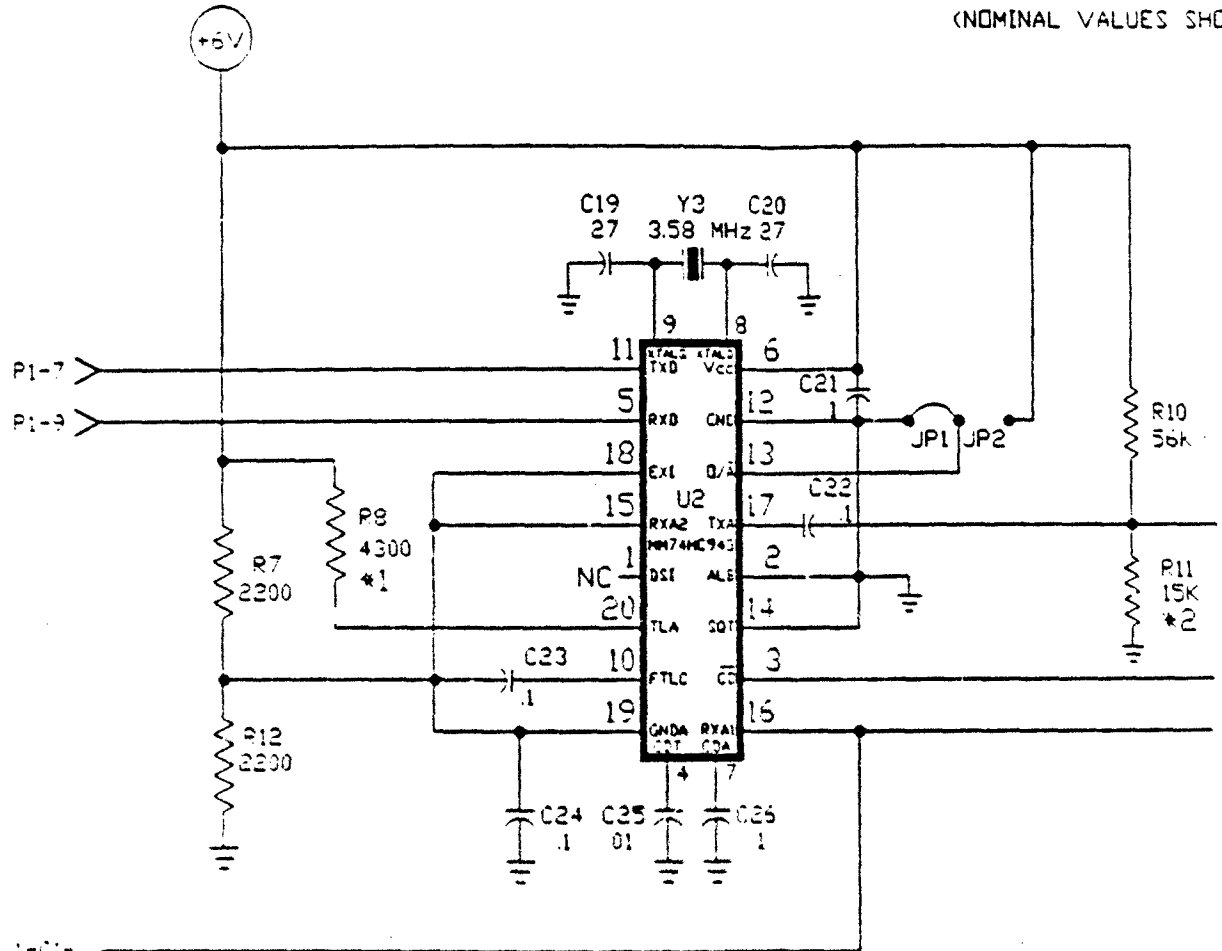


CONTRACT NO.		PURDUE UNIVERSITY	
DAMD17-87-C-7105		W. A. MILLENBRAND BIOMEDICAL ENGINEERING CENTER	
APPROVALS	DATE	WEST LAFAYETTE, IN 47907	
DRAWN	CG	PMC RADIO COMMUNICATIONS LINK SCHEMATIC	
CHECKED	JIT		
		SIZE	DRAWING NO.
		B	PMC-RAD-SCH
		SCALE	SHEET
		N/A	1 OF 3

SIZE B	FORM NO.	DRAWING NO. PMC-RAD-SCH
------------------	----------	-----------------------------------

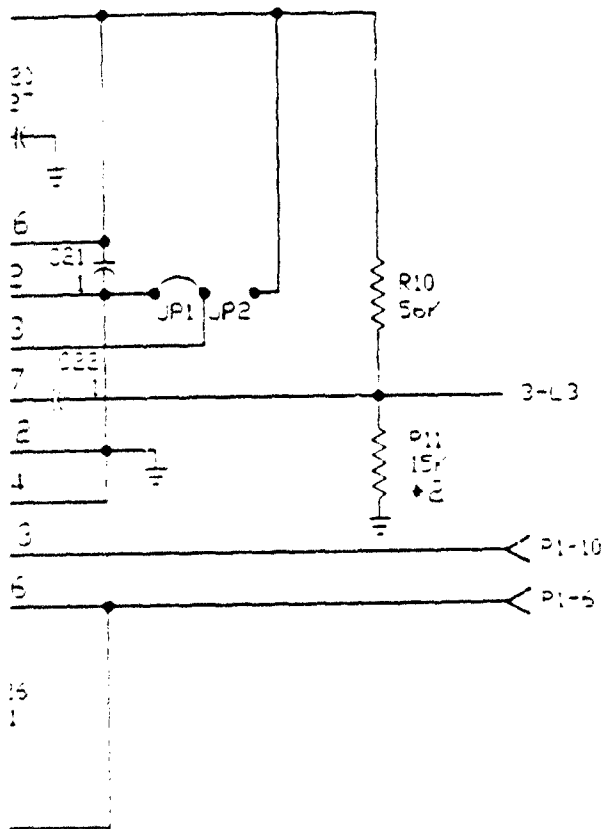
SCALE	N/A	SHEET	1	OF	3
-------	-----	-------	---	----	---

*1 Select for 4kHz Devia
 *2 Select for $F_c=72.170$
 (NOMINAL VALUES SHD)



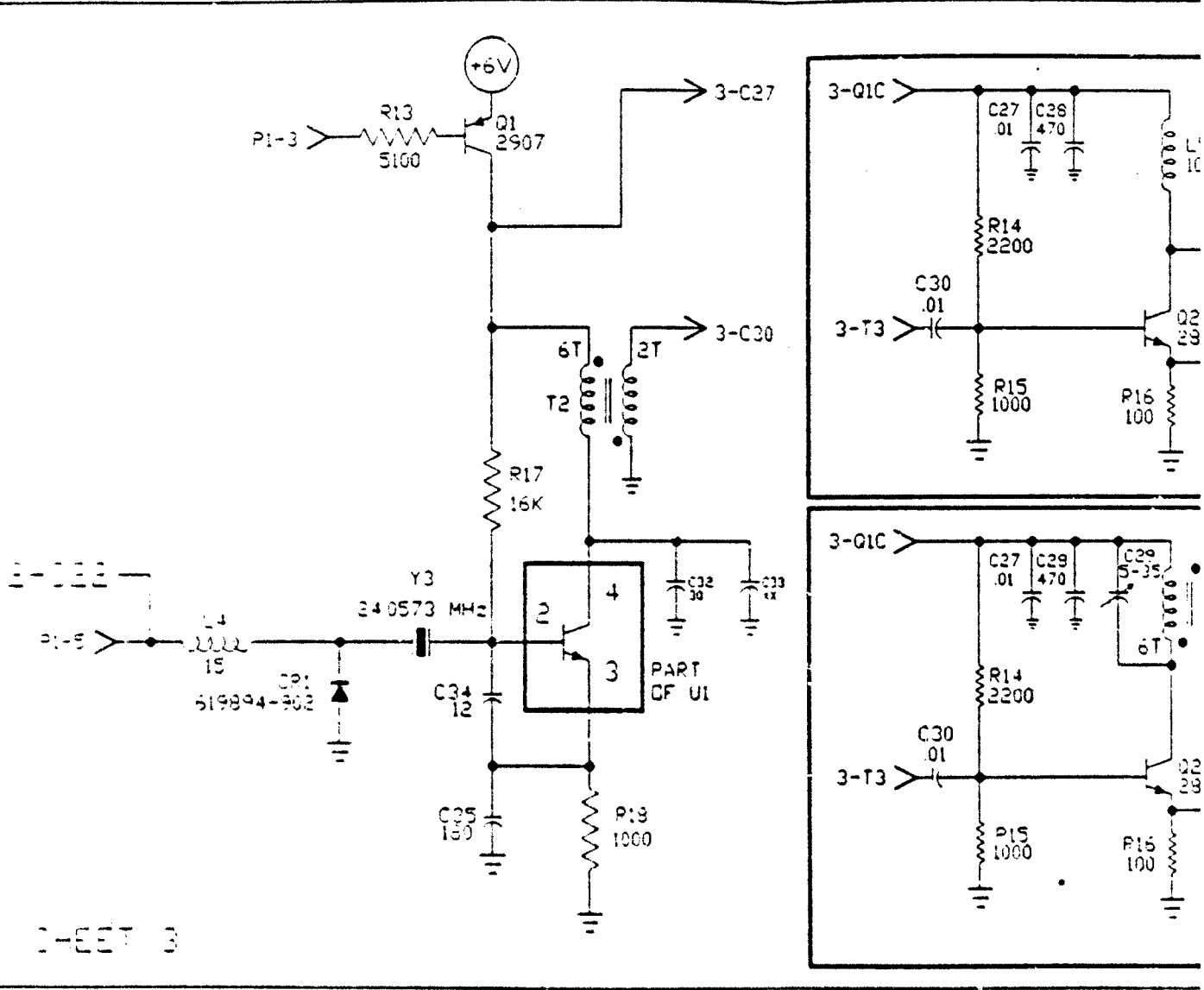
1-016
 SHEET 2

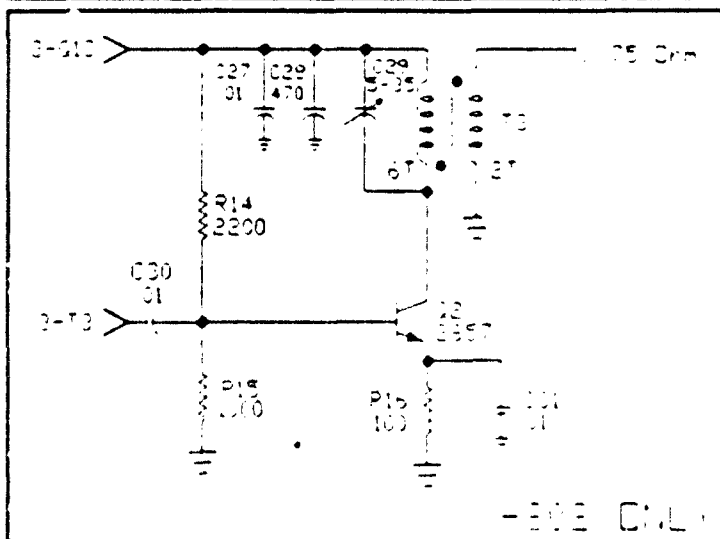
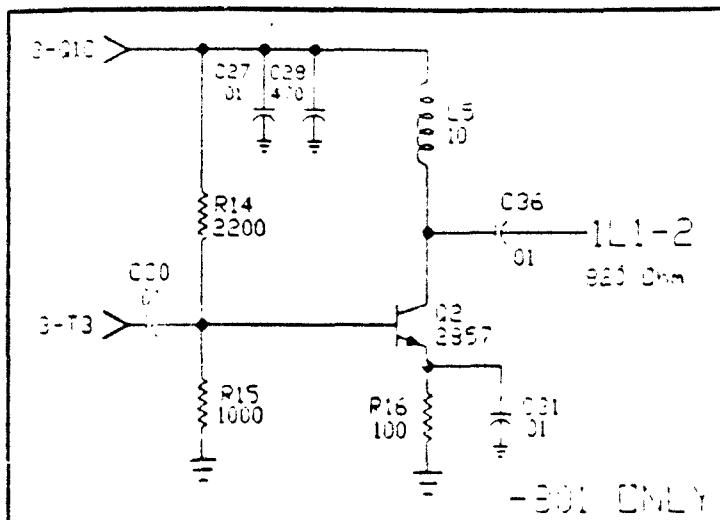
*1 Select for 4kHz Deviation +/-200Hz
*2 Select for Fc=72.170 MHz +/-25Hz
(NOMINAL VALUES SHOWN)



DOCUMENT SUPPLIED BY MAGNAVOX

SIZE B	FSCM NO.	DRAWING NO. PMC-RAD-SCH
SCALE N/A		SHEET 2 OF 3

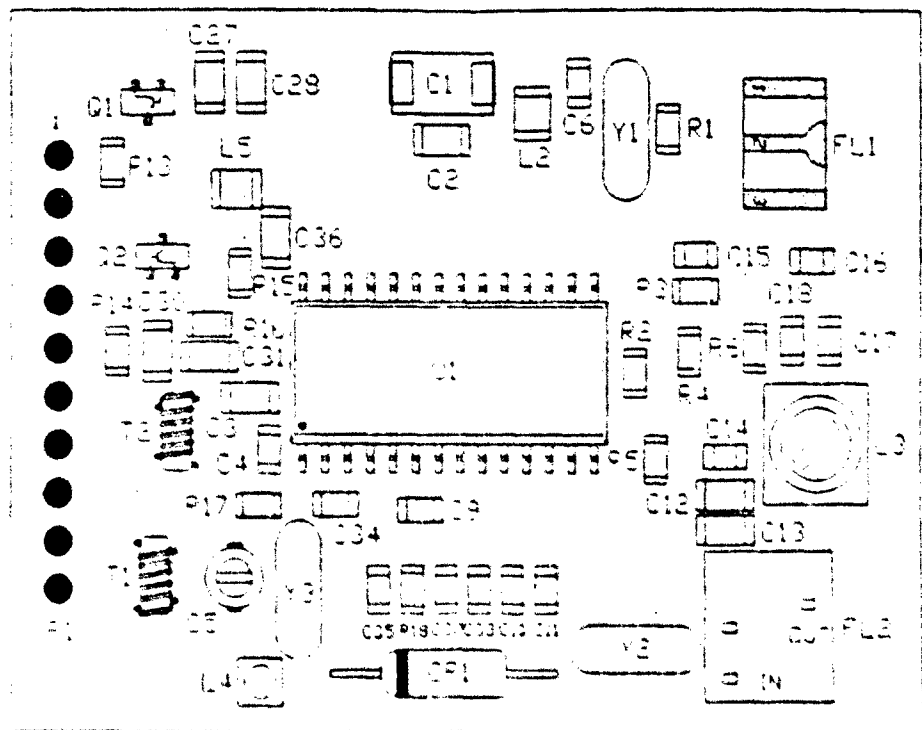




NOTES: 1. PMC RADIO LINK USES ASSY -801

DOCUMENT SUPPLIED BY MAGNAVOX

REV	FIG. NO.	DRAWING NO.
B		PMC-RAC-SCH
SCALE	N A	SHEET 3 OF 3

[illegible]

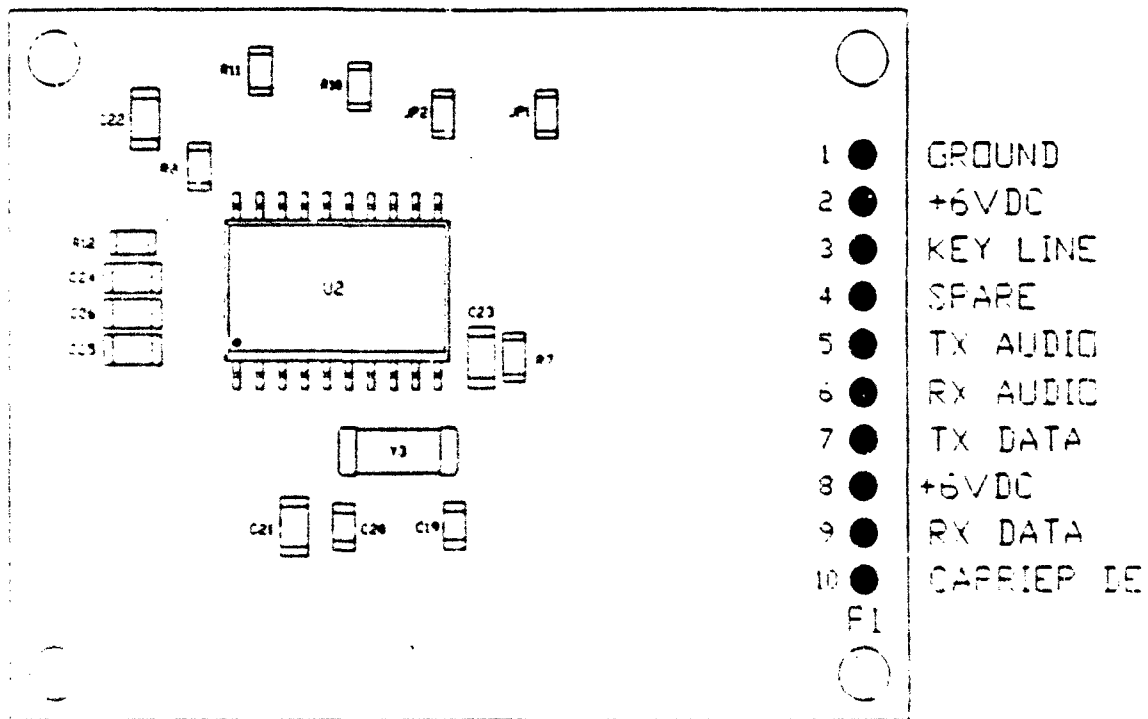
CONTRACT NO.	
DAMD17-87-C-719	
APPROVALS	
DRAWN	GG
CHECKED	STJ

1. *Staphylococcus aureus*

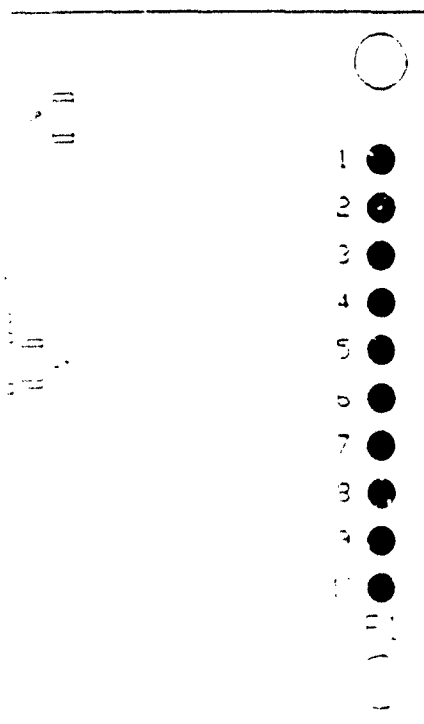


CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. MILLENBRAND STOMELYAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC RADIO COMMUNICATIONS LINK PRINTED WIRING MASTER	
DRAWN	DATE		
CHECKED	DATE	<div style="display: flex; justify-content: space-between;"> REV. B FORM NO. DRAWING NO. PMC-RAD-PWM </div> <div style="display: flex; justify-content: space-between;"> SCALE N.A. SHEET 1 OF 2 </div>	

COMPONENT PLACEMENT - REAL



LEMENT - REAF



- 1 ● GROUND
- 2 ● +5VDC
- 3 ● -5V LINE
- 4 ● ORAPE
- 5 ● TX AUDIO
- 6 ● RX AUDIO
- 7 ● TX DATA
- 8 ● +5VDC
- 9 ● RX DATA
- 10 ● CARRIER DETECT

DOCUMENT SUPPLIED BY MAGNAVOX

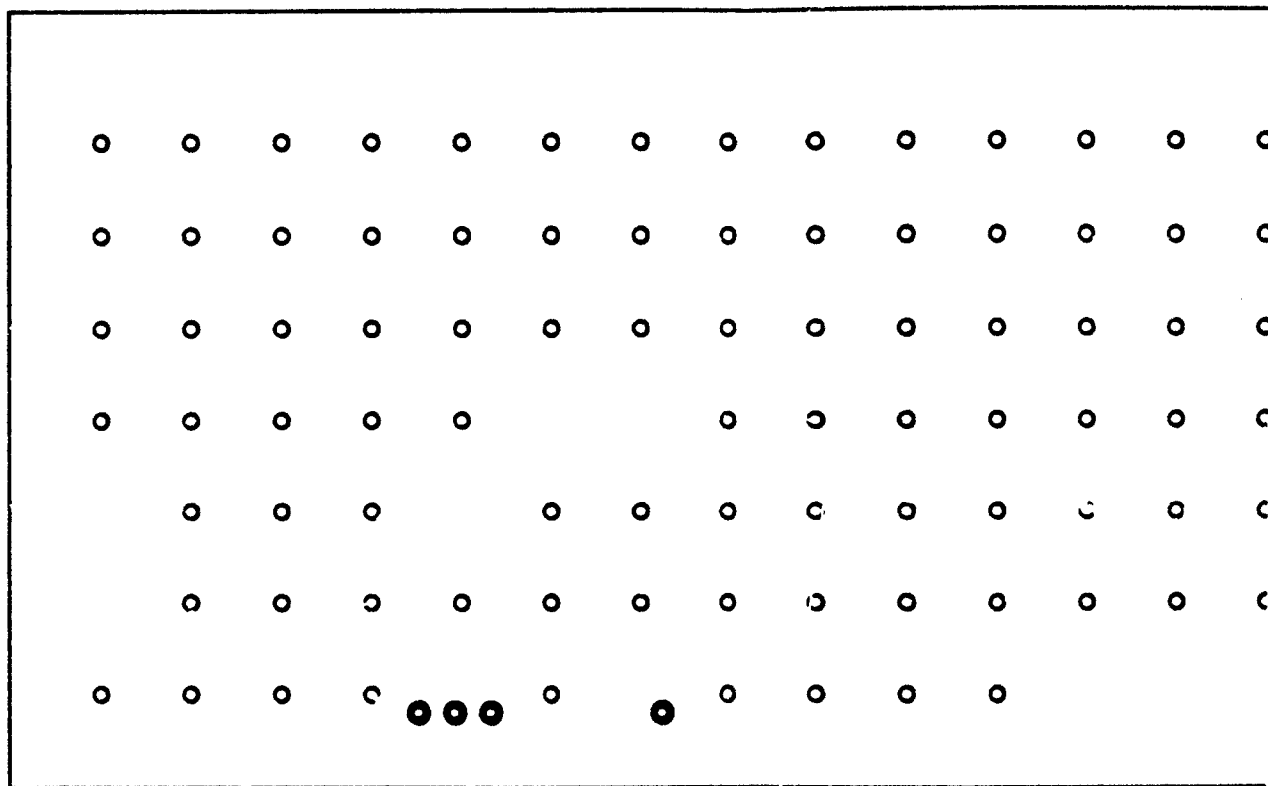
REV B	FORM NO.	DRAWING NO. PNC-RAD-PWM
SCALE NA		SHEET 2 OF 2

PARTS W. A. HILLENBRAND BIOMED. DAMD17-87-C-7195 PMC-RAD-PL 9/10/89
LIST ENGR. CENTER, PURDUE UNIV.

PMC RADIO COMMUNICATIONS LINK BOARD

SHEET 1 OF 1

COMPOSER EXTERNAL PARTS LIST FOR ASSEMBLY ES-502-3772-02-81-981				25/10/89 PAGE 1
MODULE ASSEMBLY--PURDUE-PMC DIGITAL RF				FORM REV P/L REV - A
PART NUMBER	ITEM	QTY	REF	DESCRIPTION
070-5500	1	1 PL	2	FILTER, MULTI-ELEMENT CARRIER
080-55450	2	1 PL	4	RESONATOR, CERAMIC, CHIP (MURATA)
ES-502-3772-02-81	3	1		PRINTED WIRING BOARD
ES-502-3772-02-82	4	REF		DIAGRAM, SCHEMATIC
SRM4017R1824E24P8	5	1 C	14	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	6	5 C	15, 25, 27, 29, 31, 33	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	7	2 C	6, 24	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	8	1 C	4	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	9	1 C	25	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	10	1 C	12	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	11	2 C	19, 20	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	12	1 C	22	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	13	1 C	11	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	14	1 C	29	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	15	1 C	16	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	16	1 C	9	CAPACITOR, FIXED, CERAMIC
SRM4017R1824E24P8	17	9 C	1, 2, 10, 11, 21, 22, 23, 24, 26	CAPACITOR, FIXED, CERAMIC (MURATA)
080-55450	18	1 PL	7	INDUCTOR, VARIABLE, TUNING (KEMET)
080-55450	19	1 PL	1	MICROCIRCUIT, LINEAR (MOTOROLA)
080-55450	20	1 PL	2	MICROCIRCUIT, DIGITAL (NATIONAL)
080-55450	21	1 PL	2	TRANSISTOR, RF, SURFACE MOUNT
080-55450	22	1 PL	1	TRANSISTOR, SURFACE MOUNT
080-55450	23	1 PL	1	FILTER, CERAMIC, CHIP (MURATA)
080-55450	24	1 C	1	CAPACITOR, FIXED, TANTALUM
080-55450	25	1 PL	4	INDUCTOR, FIXED, CHIP (AMERICAN PRECISION IND.)
080-55450	26	1 PL	2	INDUCTOR, FIXED, CHIP (COILCRAFT)
080-55450	27	1 PL	5	INDUCTOR, FIXED, CHIP (COILCRAFT)
080-55450	28	1 PL	15	RESISTOR, FIXED, FILM, CHIP (100 ohm)
080-55450	29	1 PL	1	RESISTOR, FIXED, FILM, CHIP (22 ohm)
080-55450	30	2 PL	13, 16	RESISTOR, FIXED, FILM, CHIP (1K ohm)
080-55450	31	2 PL	7, 10, 14	RESISTOR, FIXED, FILM, CHIP (1.5K ohm)
080-55450	32	1 PL	3	RESISTOR, FIXED, FILM, CHIP (5.1K ohm)
080-55450	33	2 PL	7, 10	RESISTOR, FIXED, FILM, CHIP (5.1K ohm)
080-55450	34	1 PL	1	RESISTOR, FIXED, FILM, CHIP (3.0K ohm)
080-55450	35	2 PL	2, 5	RESISTOR, FIXED, FILM, CHIP (10K ohm)
080-55450	36	1 PL	17	RESISTOR, FIXED, FILM, CHIP (10K ohm)
080-55450	37	1 PL	6	RESISTOR, FIXED, FILM, CHIP (20K ohm)
080-55450	38	1 PL	10	RESISTOR, FIXED, FILM, CHIP (50K ohm)
080-55450	39	1 PL	6	RESISTOR, FIXED, FILM, CHIP (50K ohm)
080-55450	40	1 PL	1	CAPACITOR, FIXED (47 pF)
080-55450	41	1 C	9	CAPACITOR, VARIABLE (0.5-20 pF)
080-55450	42	1 PL	1	SEMICONDUCTOR, DIODE, VARIACAP
080-55450	43	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	44	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	45	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	46	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	47	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	48	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	49	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)
080-55450	50	1 PL	1	CRYSTAL, QUARTZ, 45.5 MHz (400 kHz)



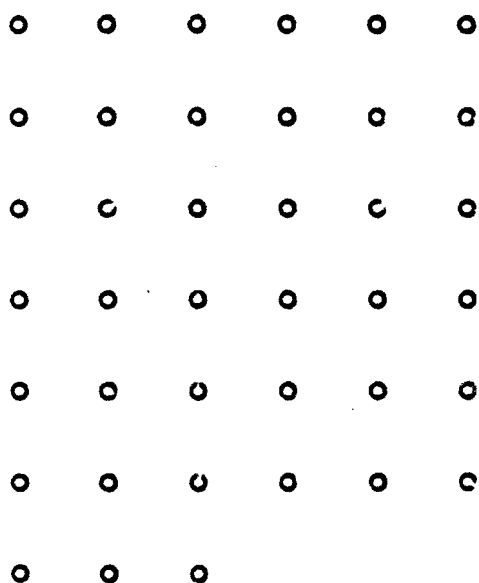
GND
PXRDET
PXPWR

V-



REDUCE TO 1750 MICES

CONTRACT NO.	
DAMD17-87-C-7	
APPROVALS	
DRAWN	
CHECKED	



RF COM LINK

GND

RPOWER

KEY

TXD

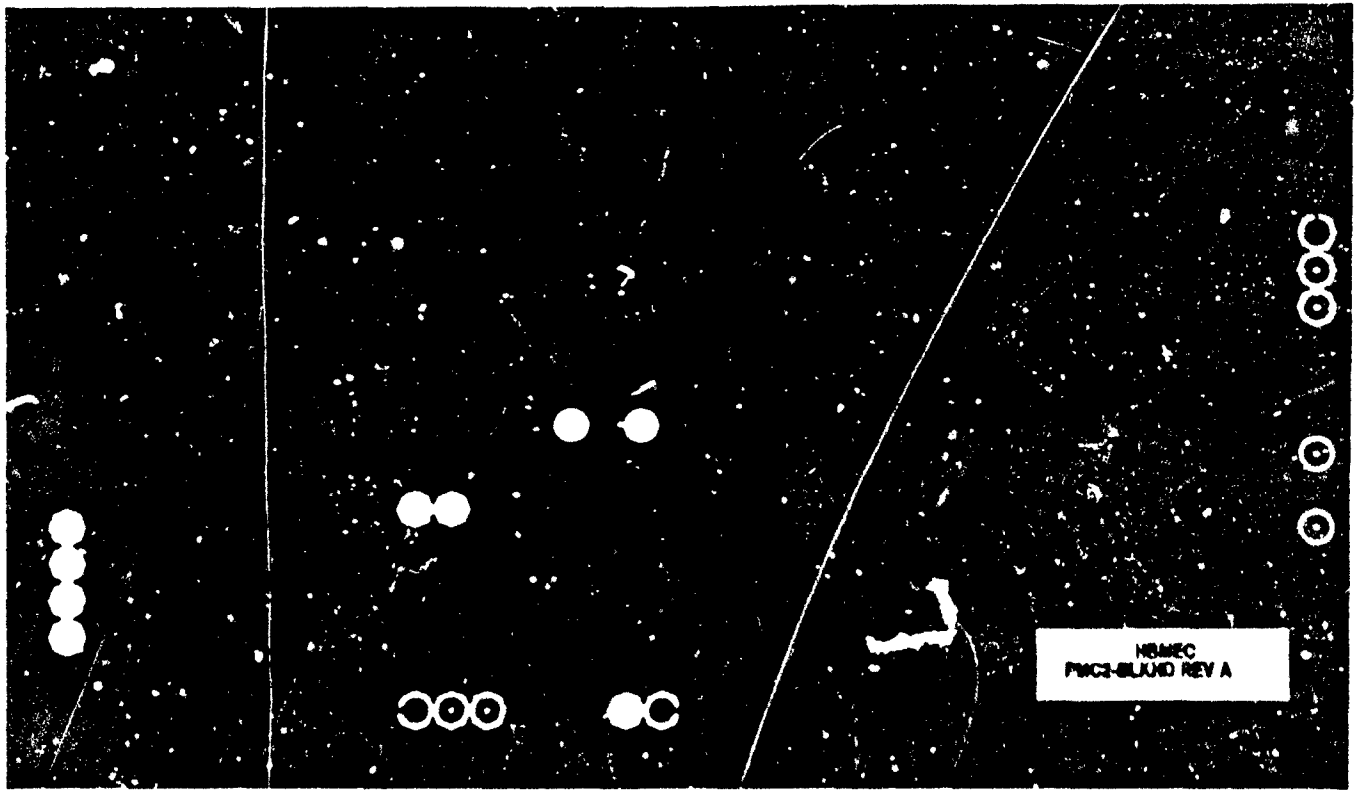
RXD

JES



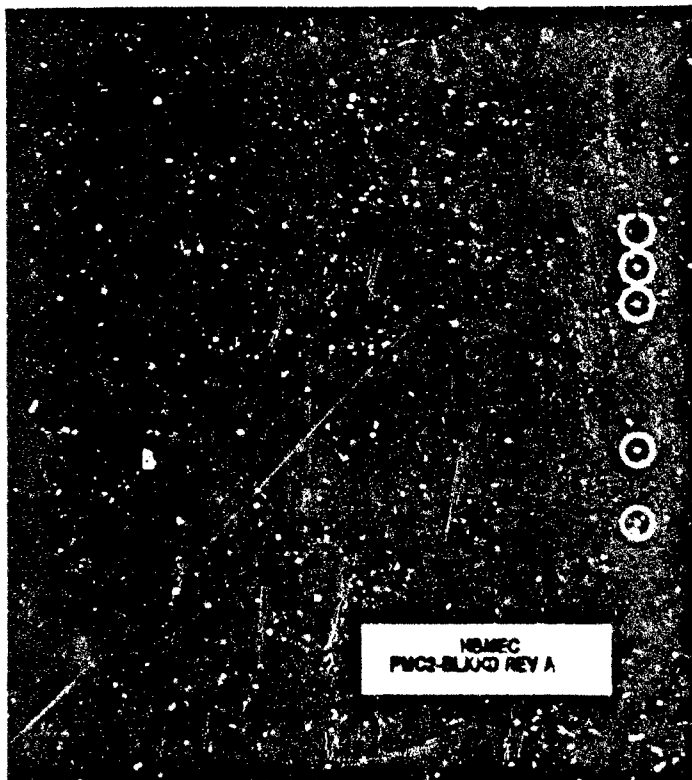
PARTS PLACEMENT

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	PMC BULKHEAD	
DRAWN GD	3-10-89		
CHECKED JTS	4-6-89	PRINTED WIRING MASTER	
		SIZE B	DRAWING NO. PMC-BHU-PWM
		SCALE 2/1	SHEET 1 OF 6



COMPONENT

REDUCE TO 1.750 INCHES



MSMEC
PMC3-BLJGD REV A

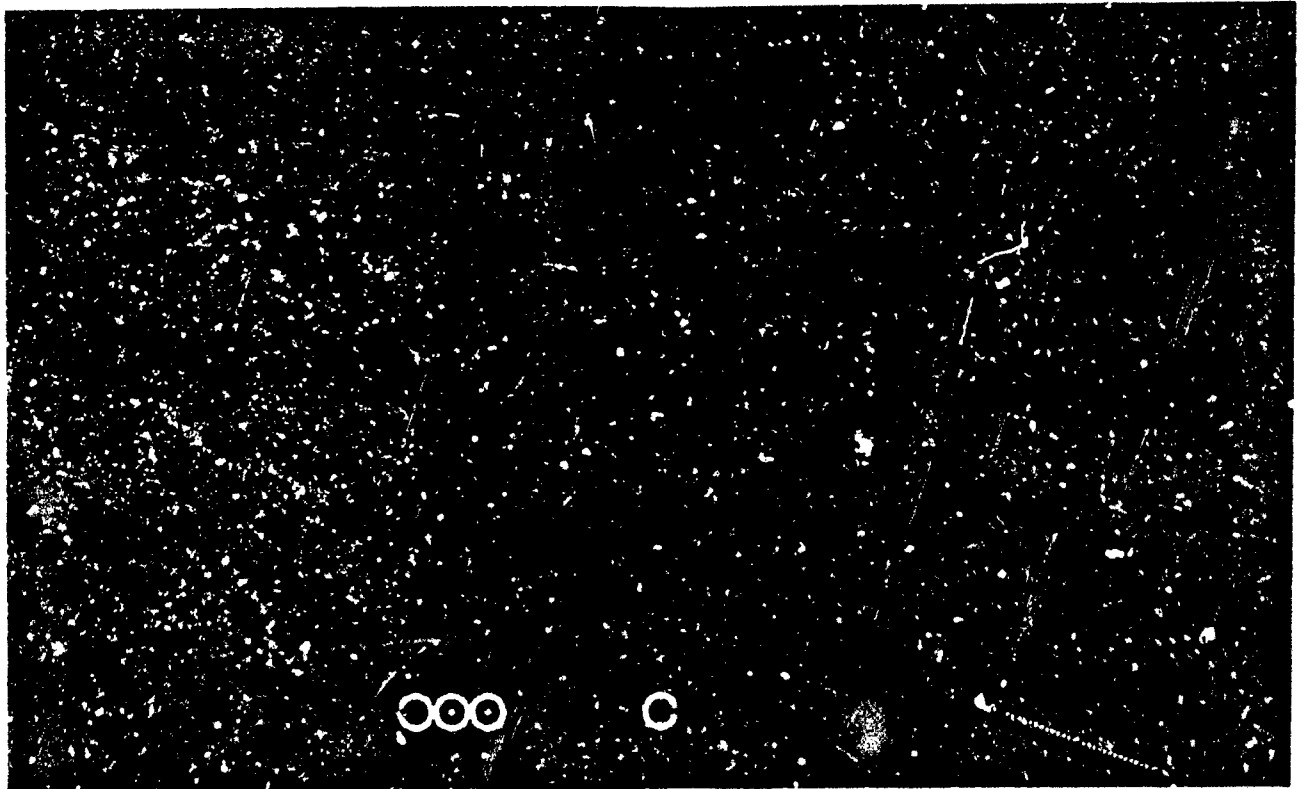
COMPONENT

.750 INCHES



TOP (COMPONENT) LEVEL

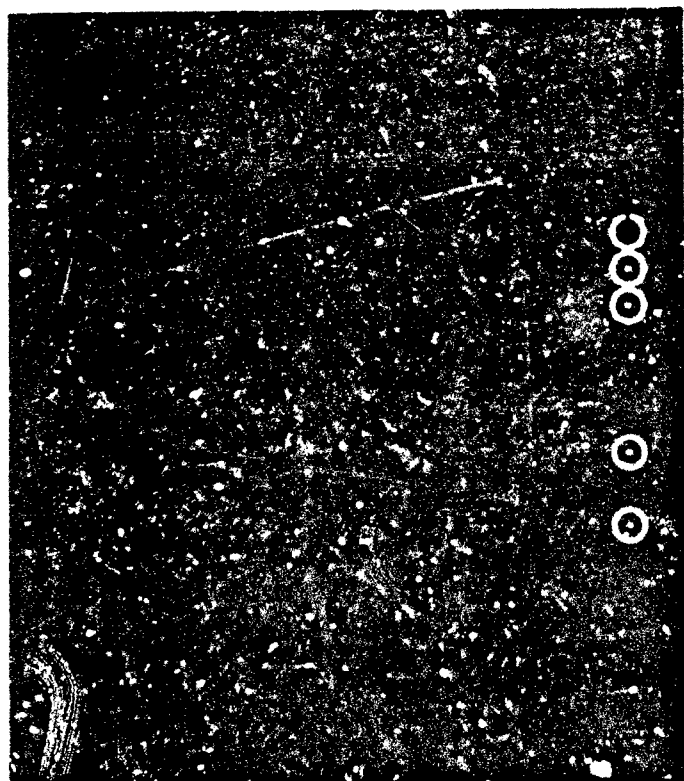
SIZE B	ITEM NO.	DRAWING NO. PMC-BHD-PWM
SCALE 2/1		SHEET 2 OF 6



MOTTO



REDUCE TO 3.750 INCHES



BOTTOM

750 INCHES



BOTTOM (SOLDER) LAYER

SIZE B	ESD NO.	DRAWING NO. PMC-EHD-PWM
SCALE 2/1		SHEET 3 OF 6

HBMEC
PAC2-BLKND REV A

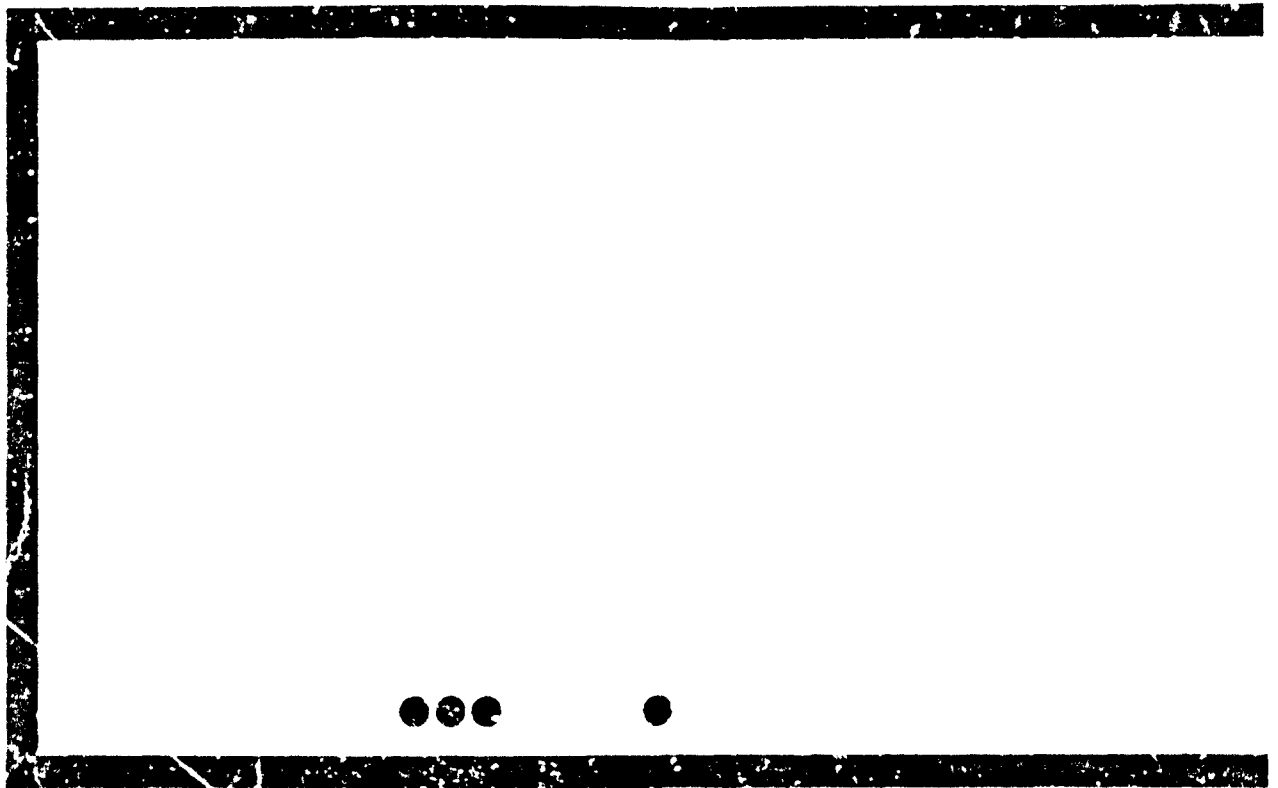
REDUCE TO 1.750 INCHES

HBMEC
PMC2-BLANK REV A

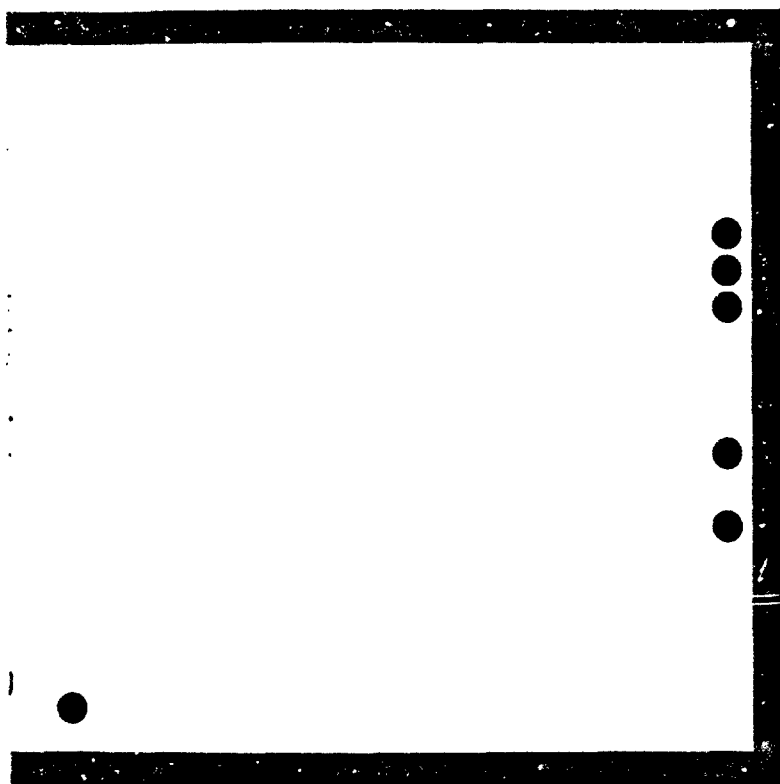
10 INCHES

SILK SCREEN LAYER

SET B	FORM NO.	DRAWING NO. PMC-BHD-PWM
SCALE 2/1		SHEET 4 OF 6



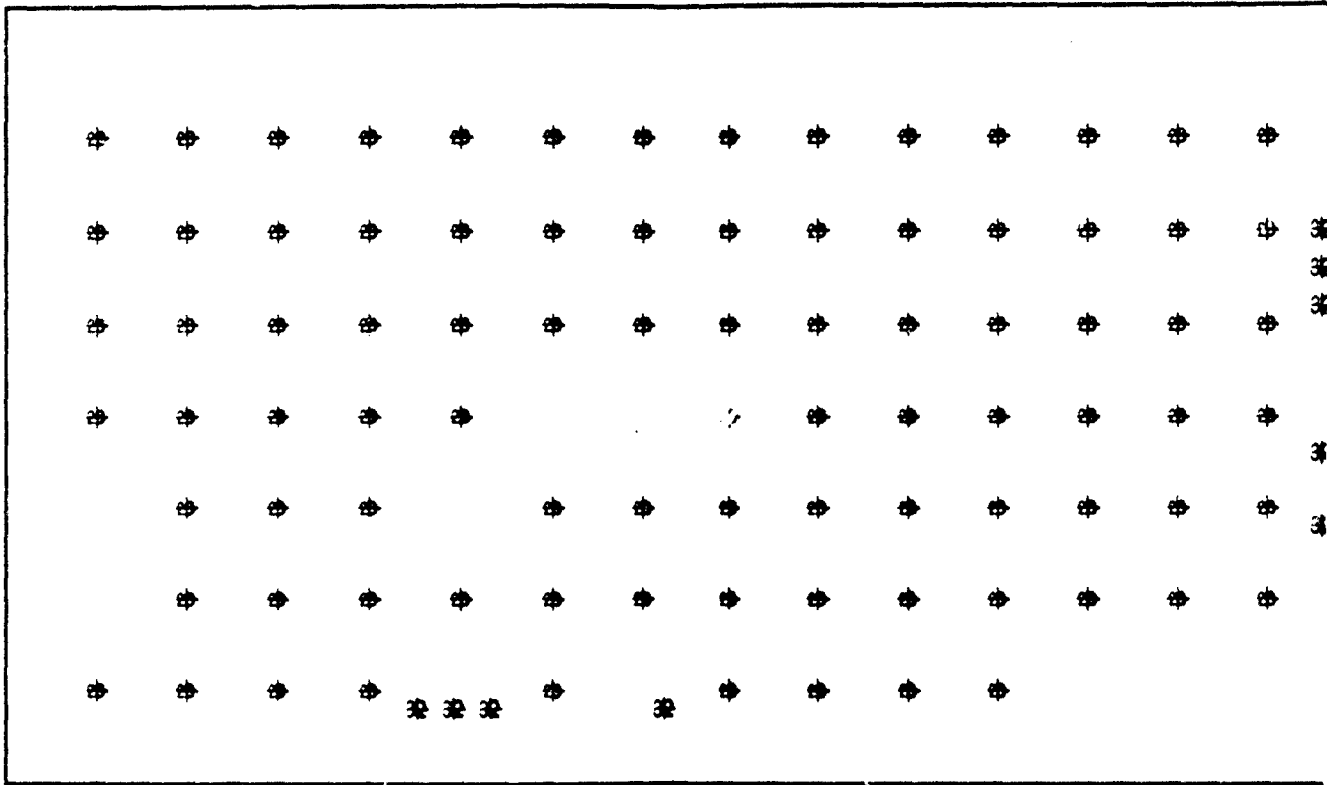
REDUCE TO 3.750 INCHES



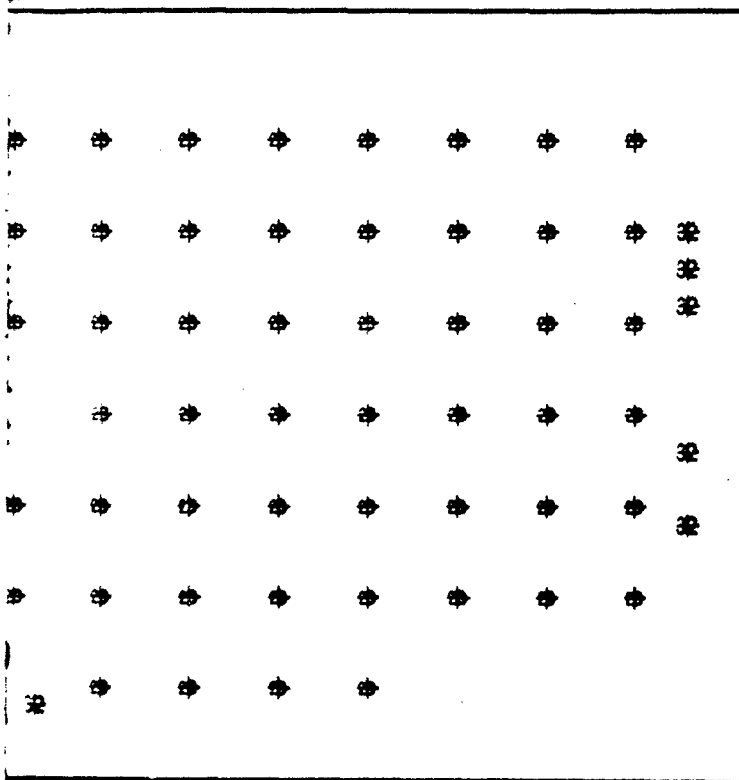
CE TO 3.750 INCHES

SOLDER MASK LAYER

SIZE B	FSCM NO.	DRAWING NO. PMC-8HD-PWM
SCALE 2/1		SHEET 5 OF 6



REDUCE TO 1.750 INCHES

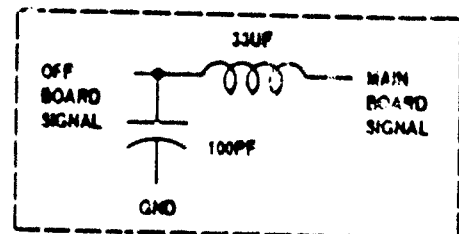
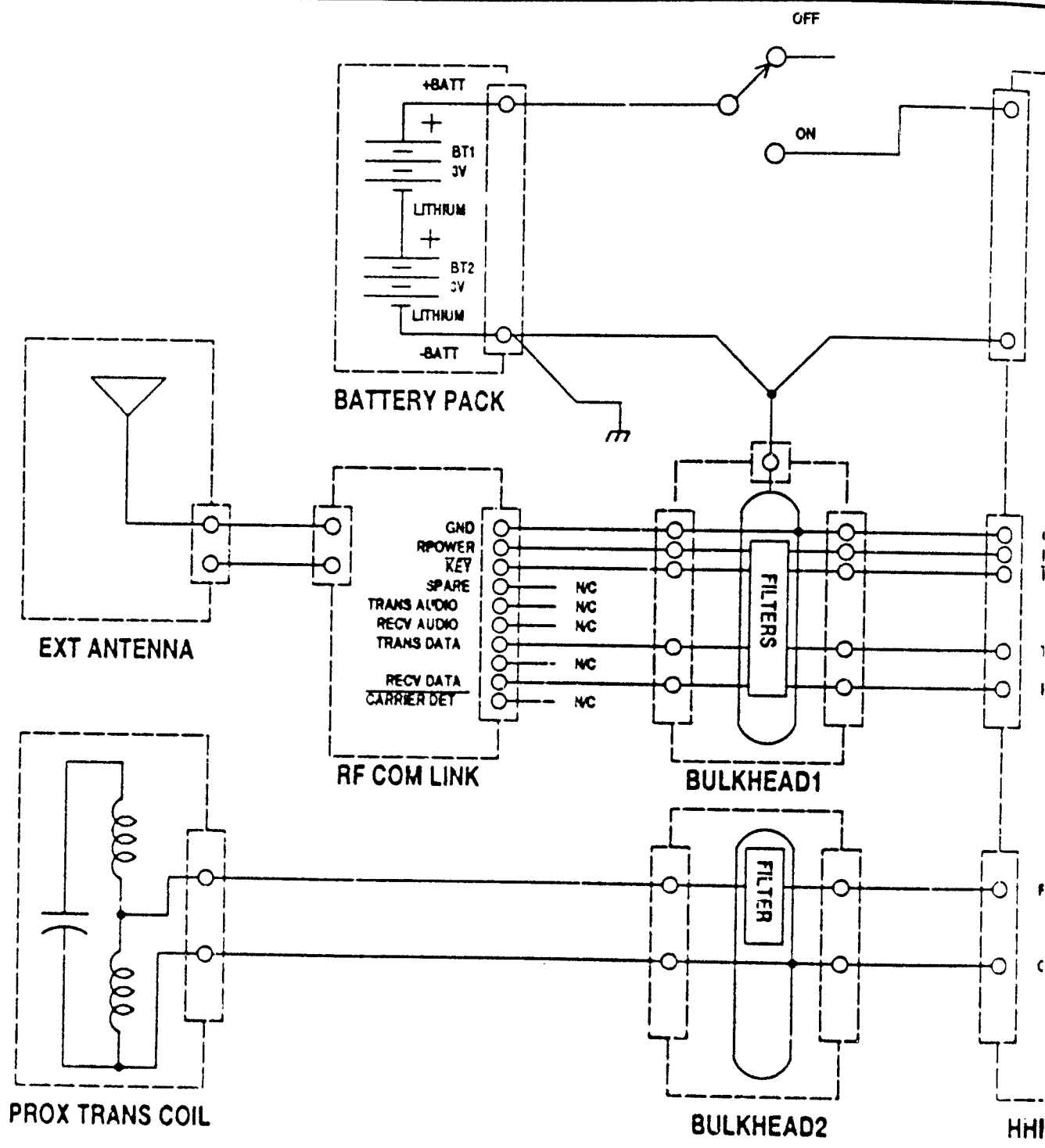


TO 1.750 INCHES

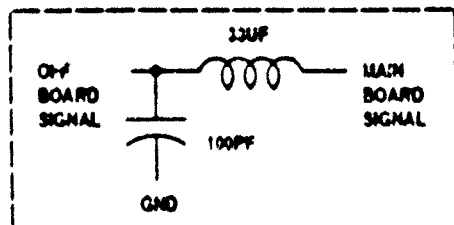
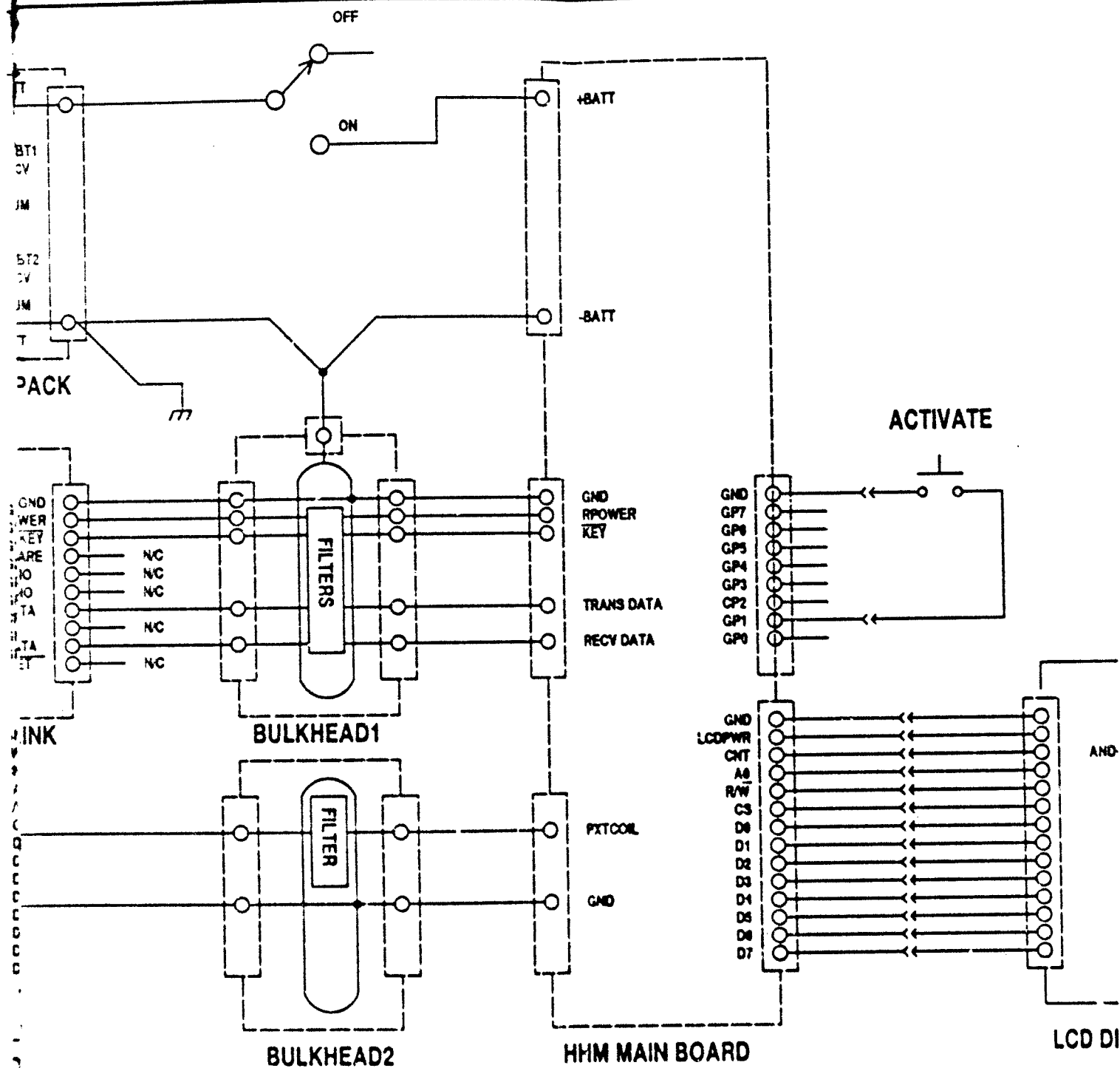


DRILL MASTER LAYER

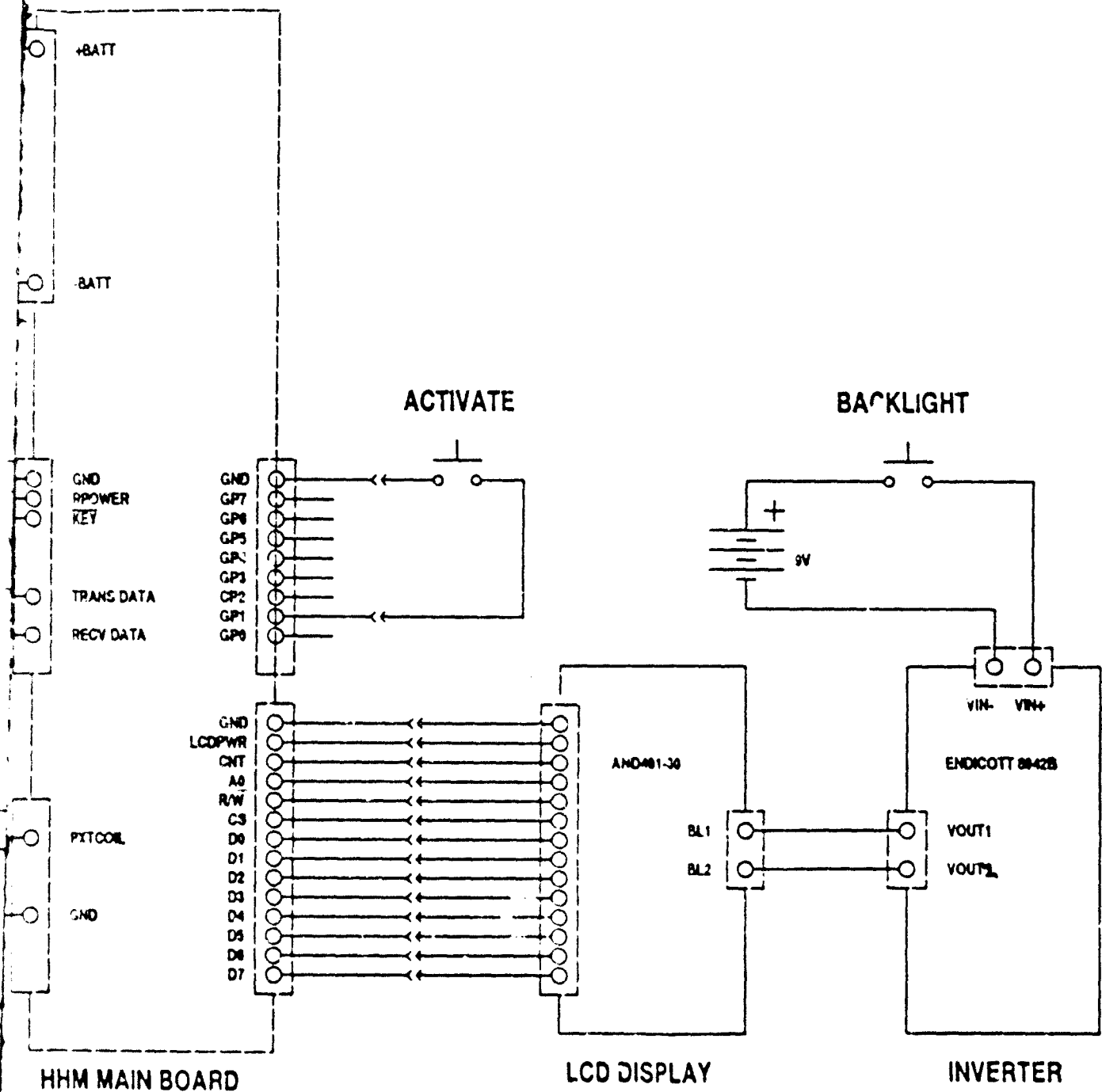
SIZE B	73CM NO.	DRAWING NO. PMC-BHD-PWM
SCALE 2/1		SHEET 6 OF 6



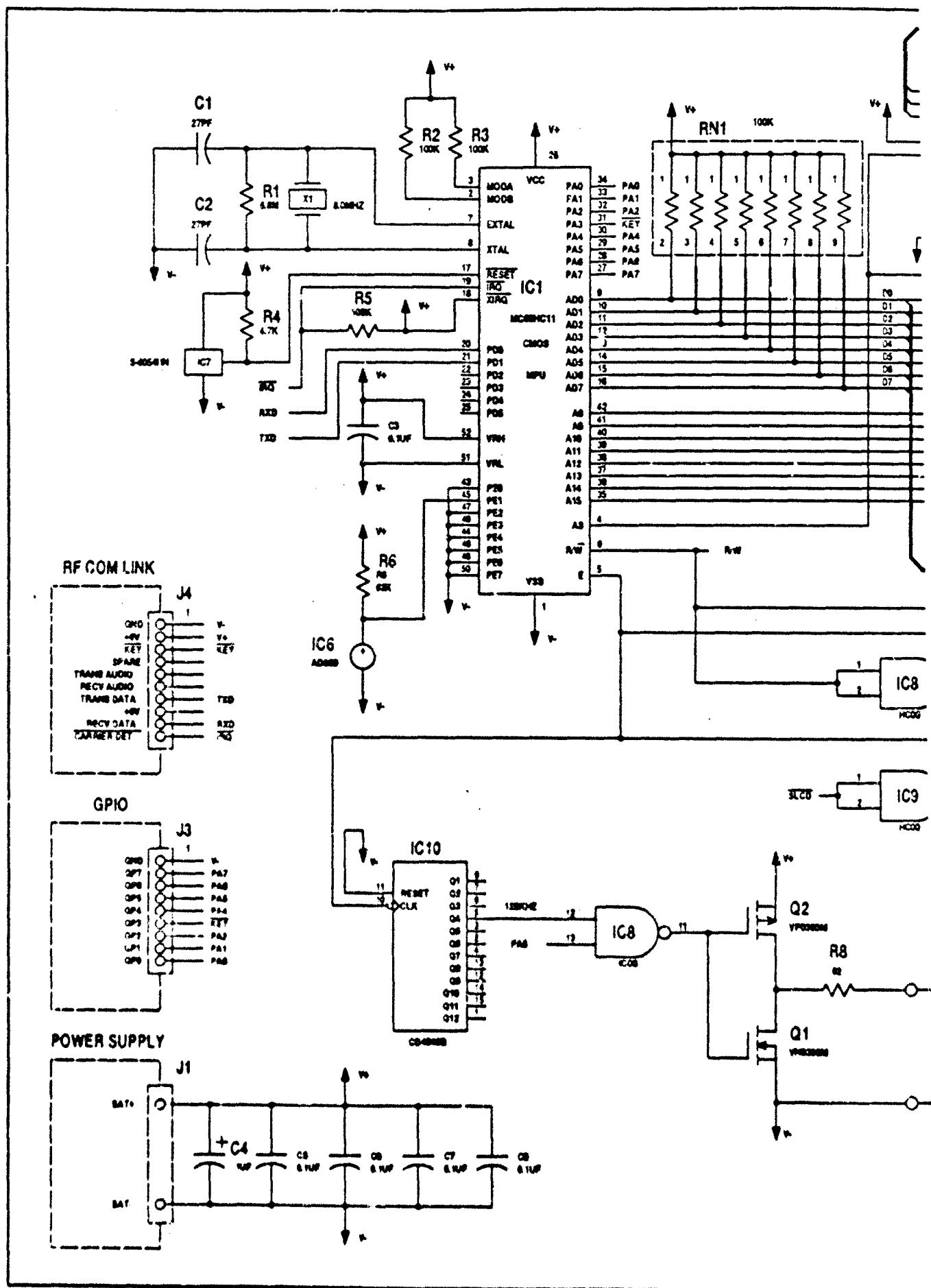
RF FILTER
(EACH LINE)

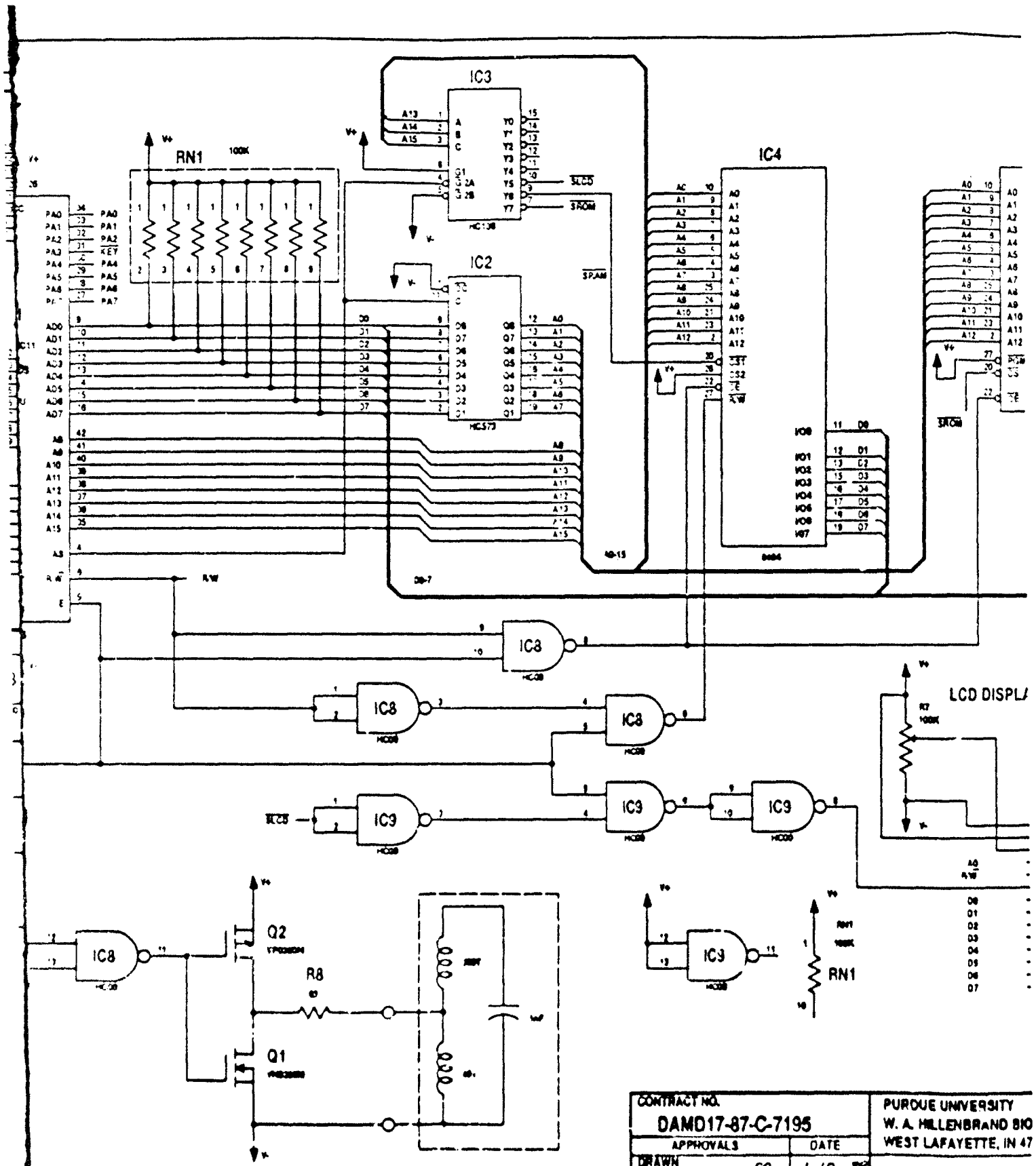


CONTRACT NO.		PURDUE
DAMD17-87-C-7195		W. A. H
APPROVALS	DATE	WEST L
DRAWN	GG	11-2-
CHECKED	JES	11-2-87
		SIZE
		8
		SCALE



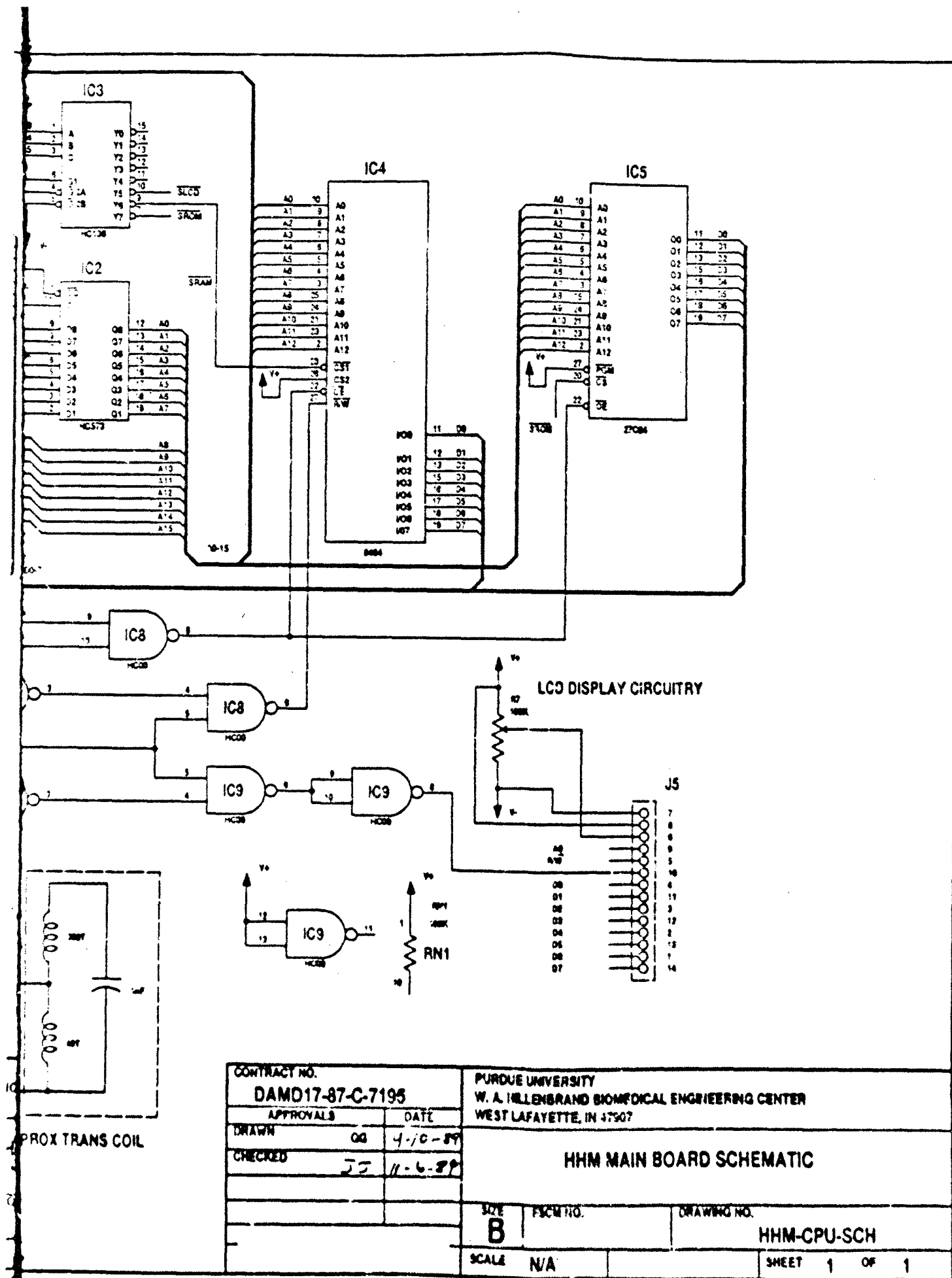
CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	HHM INTERCONNECTION DIAGRAM	
DRAWN <i>OG</i>	<i>11-1-87</i>		
CHECKED <i>OG</i>	<i>11-1-87</i>		
		SIZE B	FSCM NO.
		DRAWING NO. HHM-ICN-SCH	
		SCALE N/A	SHEET 1 OF 1





CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND Bldg WEST LAFAYETTE, IN 47	
APPROVALS		DATE	
DRAWN	GA	4-10-87	
CHECKED	JS	11-6-89	
		SIZE B	FORM NO.
		SCALE	N/A

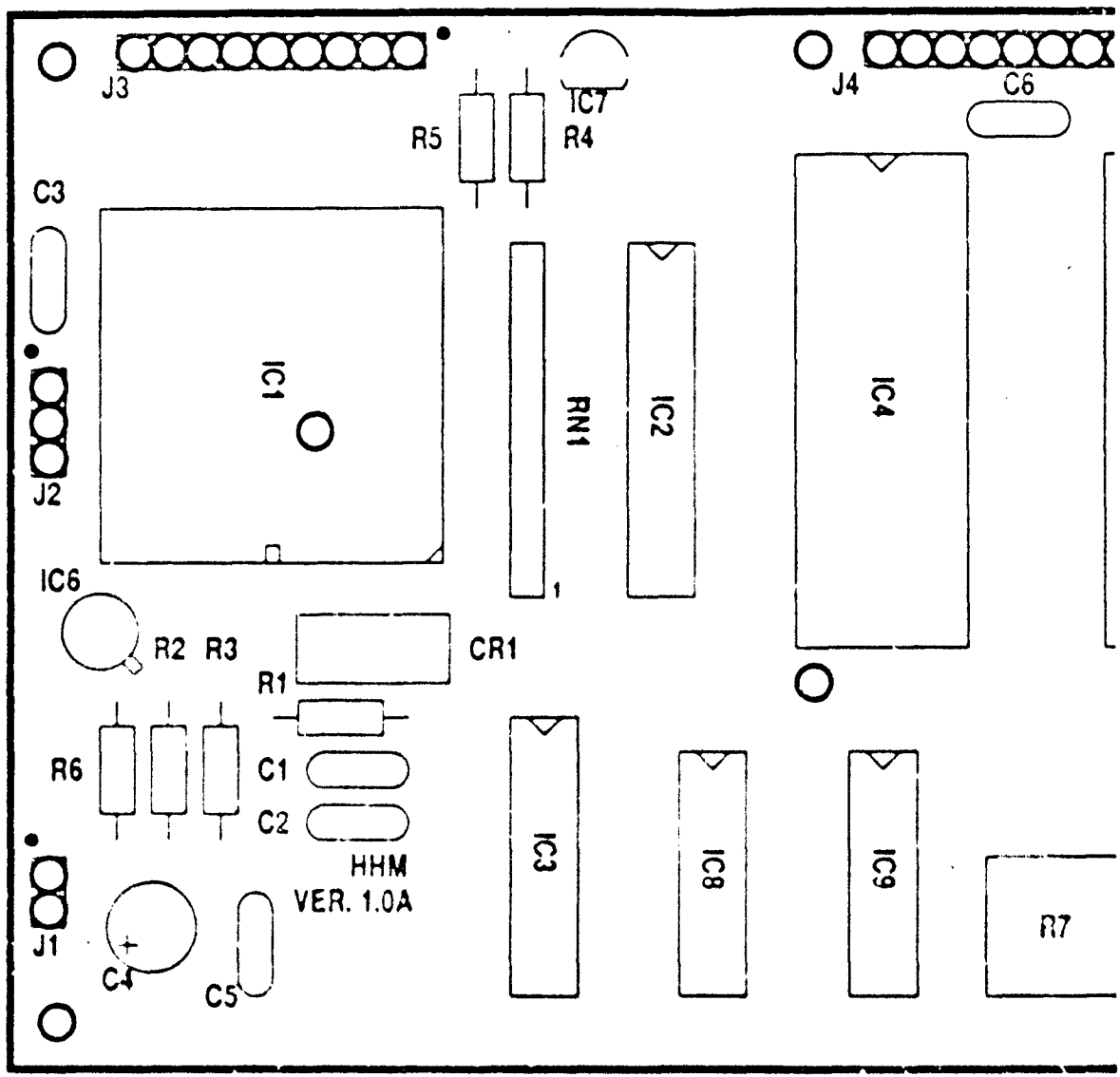
HHM A





4.0 INCHES

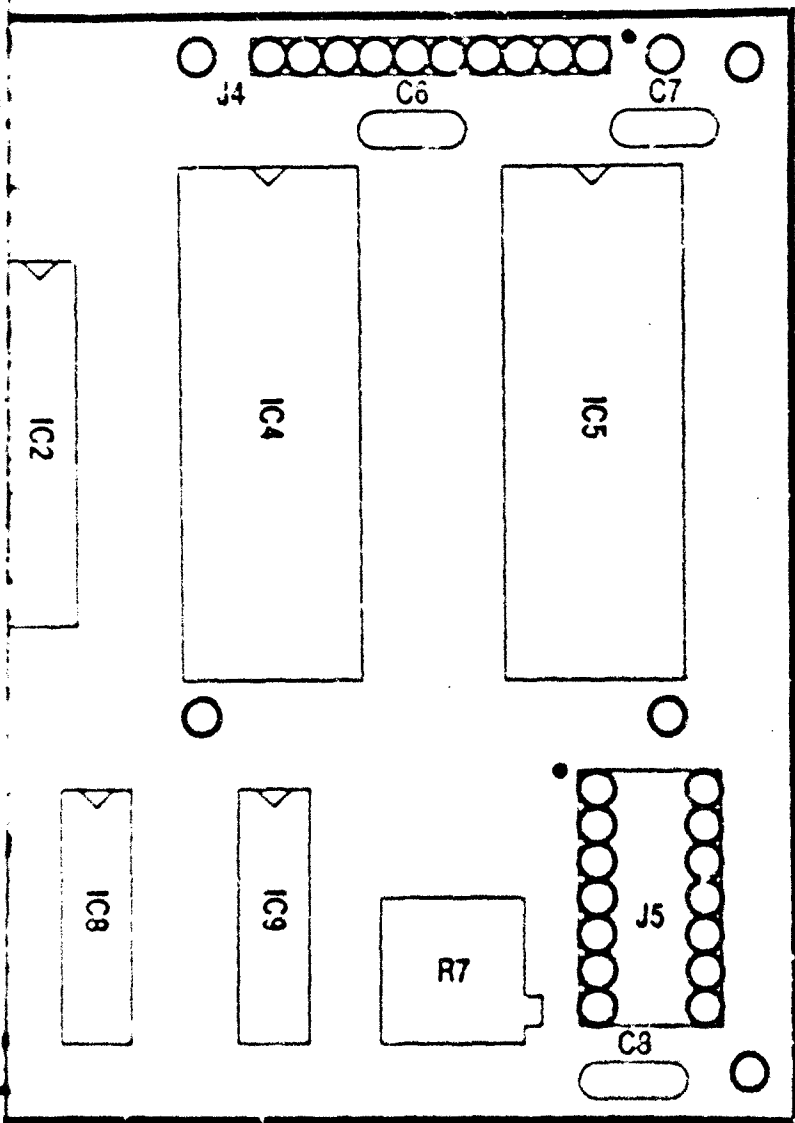
REDUCE TO THIS DIMENSION



CONTRACT NO.
DAMD17-4
APPROVALS
DRAWN
CHECKED

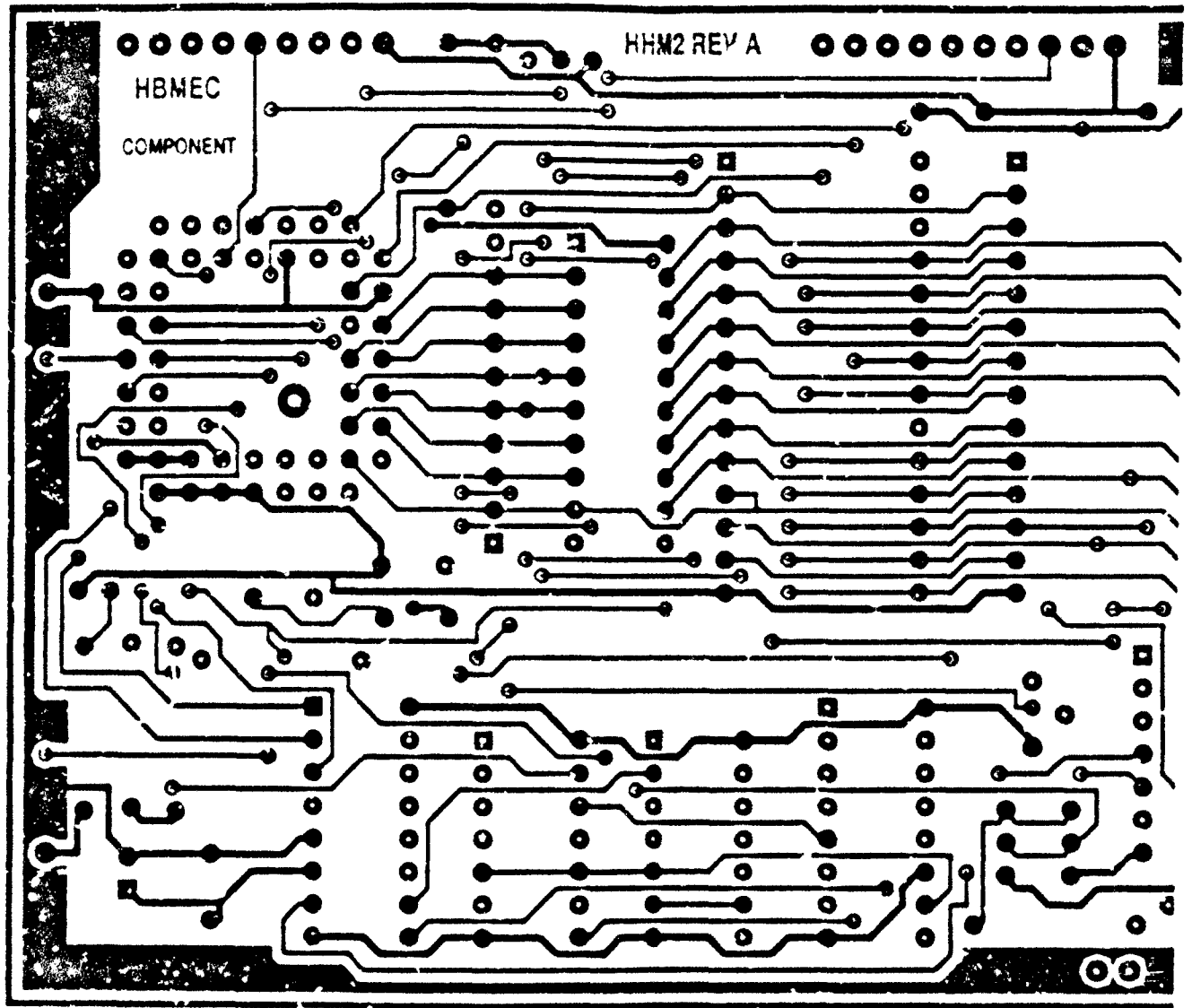
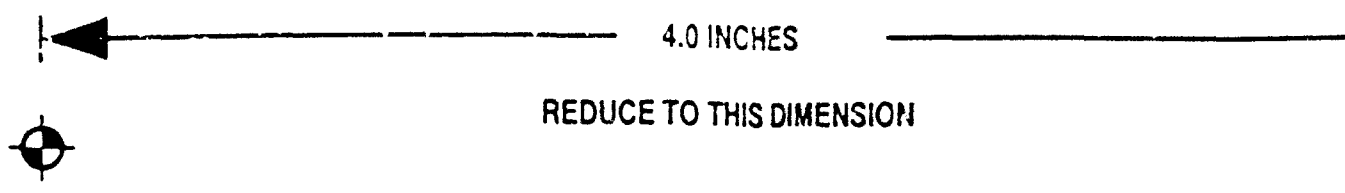
INCHES

THIS DIMENSION



PARTS PLACEMENT

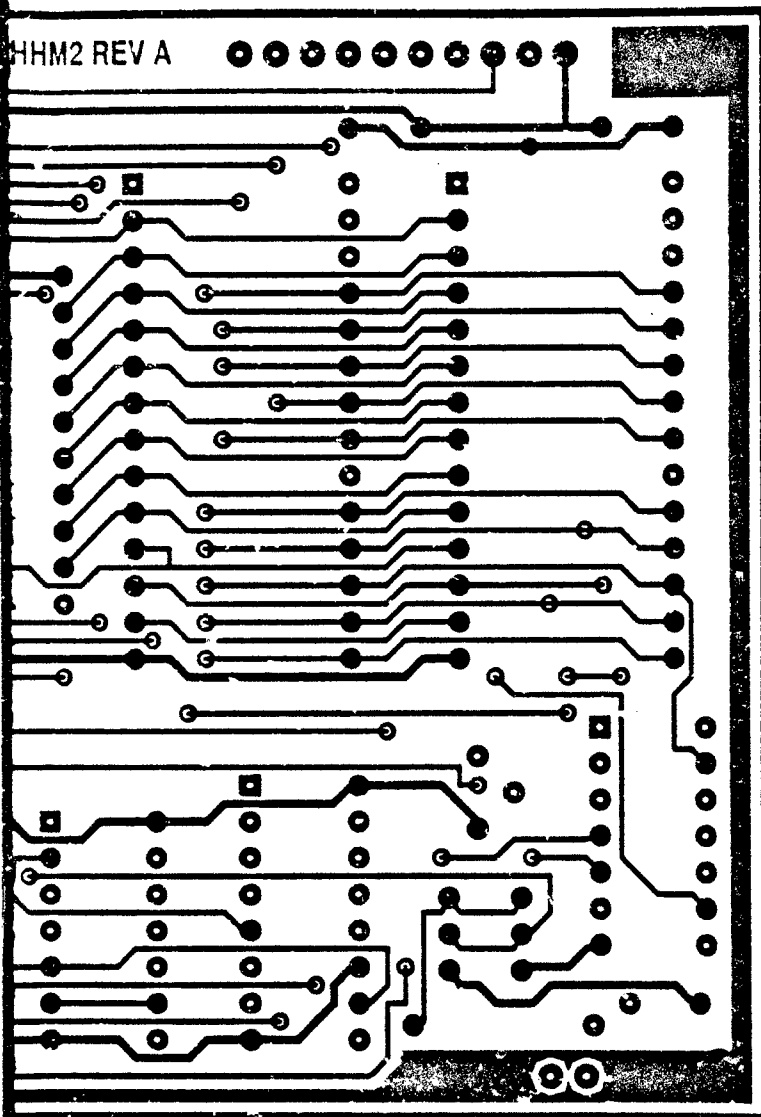
CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	HHM MAIN BOARD PRINTED WIRING MASTER	
DRAWN <i>GG</i>	<i>9-10-89</i>		
CHECKED <i>DTJ</i>	<i>11-1-89</i>	DRAWING NO. HHM-CPU-PWM	
SIZE B		PB CH NO.	SHEET 1 OF 6
SCALE 2/1			



INCHES

THIS DIMENSION

HHM2 REV A



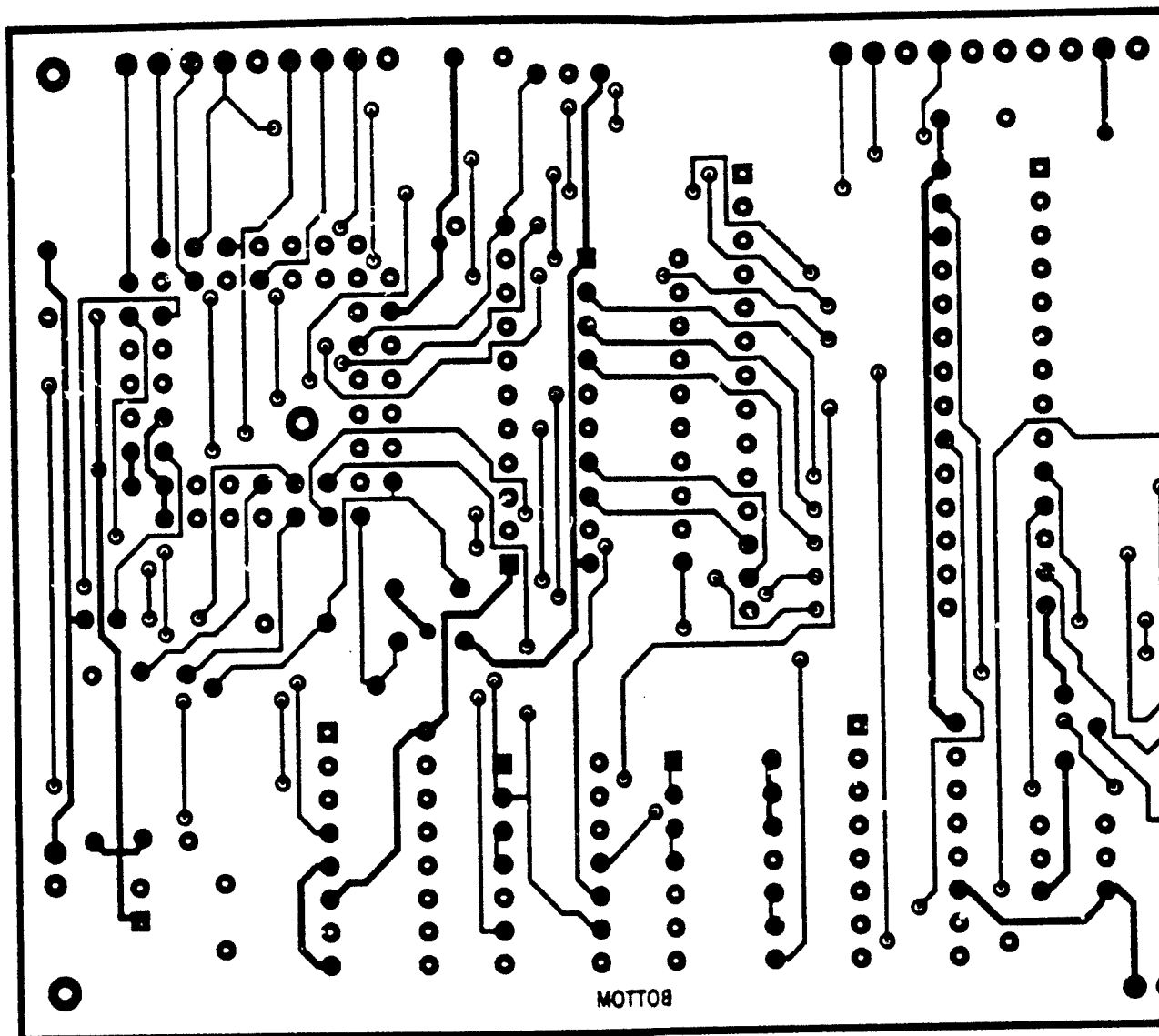
TOP (COMPONENT) LEVEL

SIZE B	FSCM NO.	DRAWING NO. HHM-CPU-PWM
SCALE 2/1		SHEET 2 OF 6



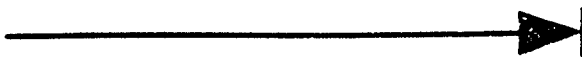
4.0 INCHES

REDUCE TO THIS DIMENSION

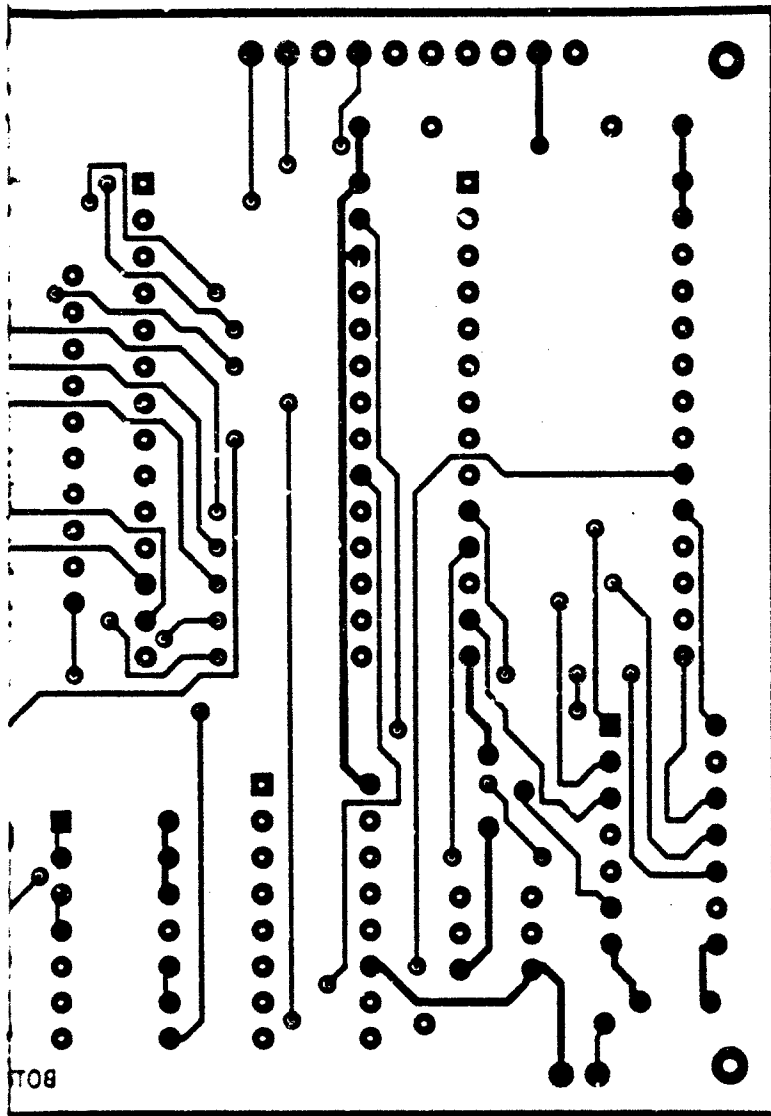


MOTTO8

INCHES



THIS DIMENSION

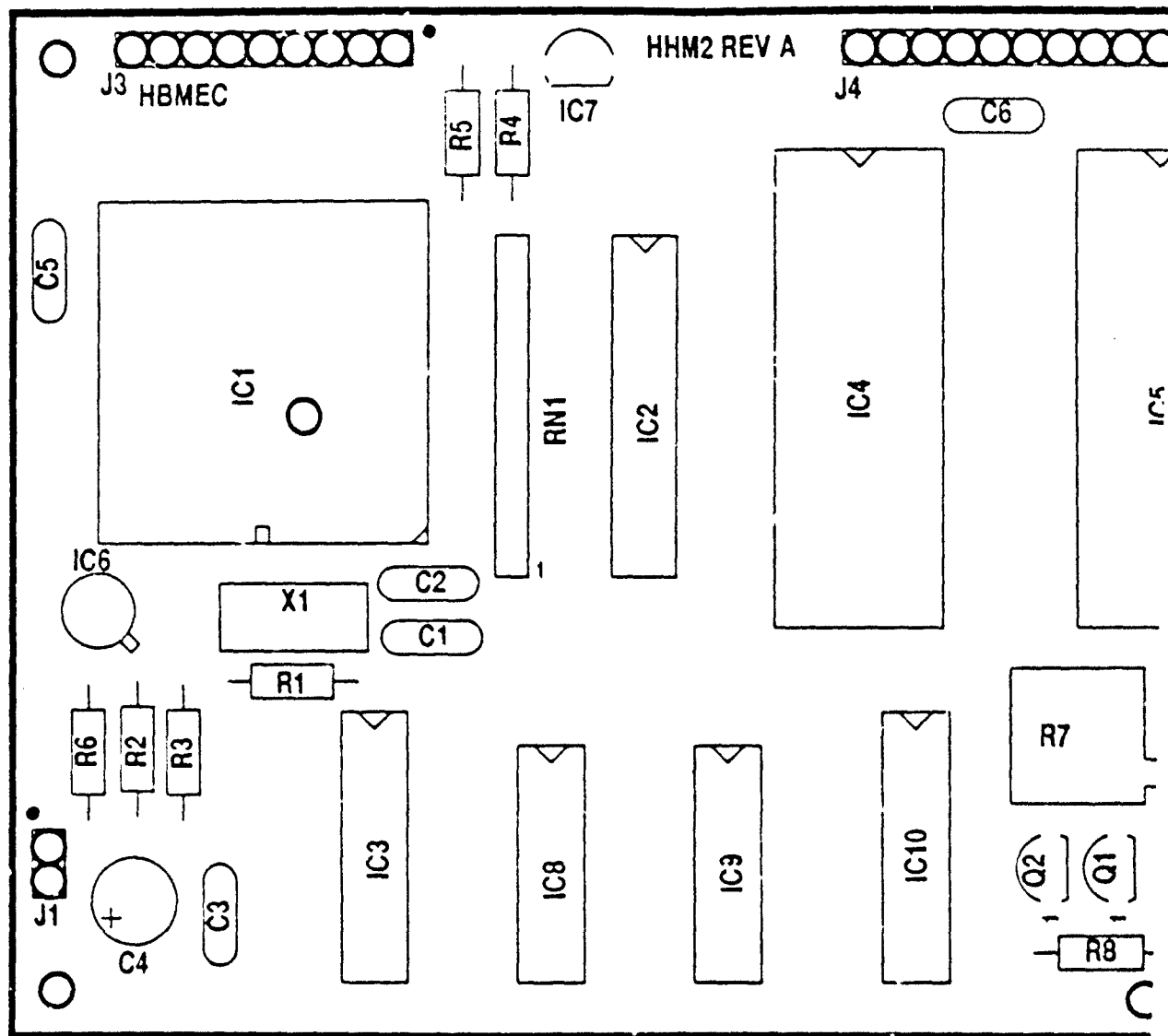


BOTTOM (SOLDER) LAYER

SIZE B	FSCM NO.	DRAWING NO. HHM-CPU-PWM
SCALE 2/1		SHEET 3 OF 6

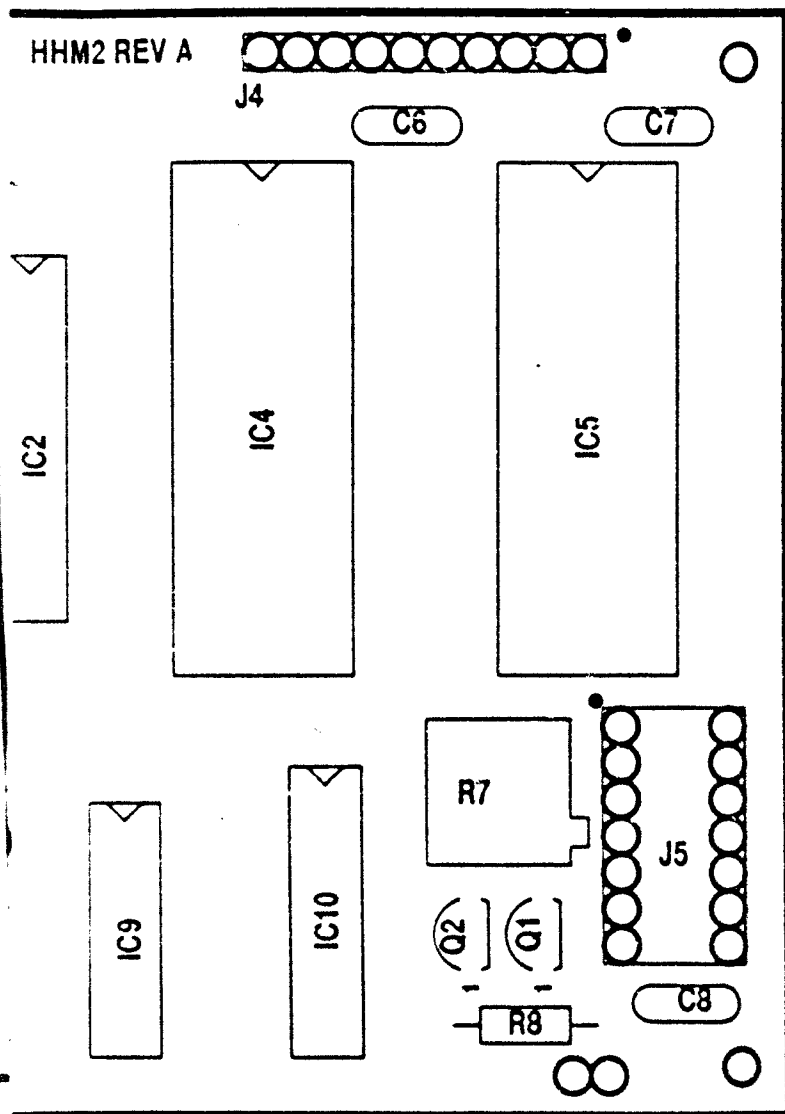
4.0 INCHES

REDUCE TO THIS DIMENSION



INCHES

THIS DIMENSION



SILK SCREEN LAYER

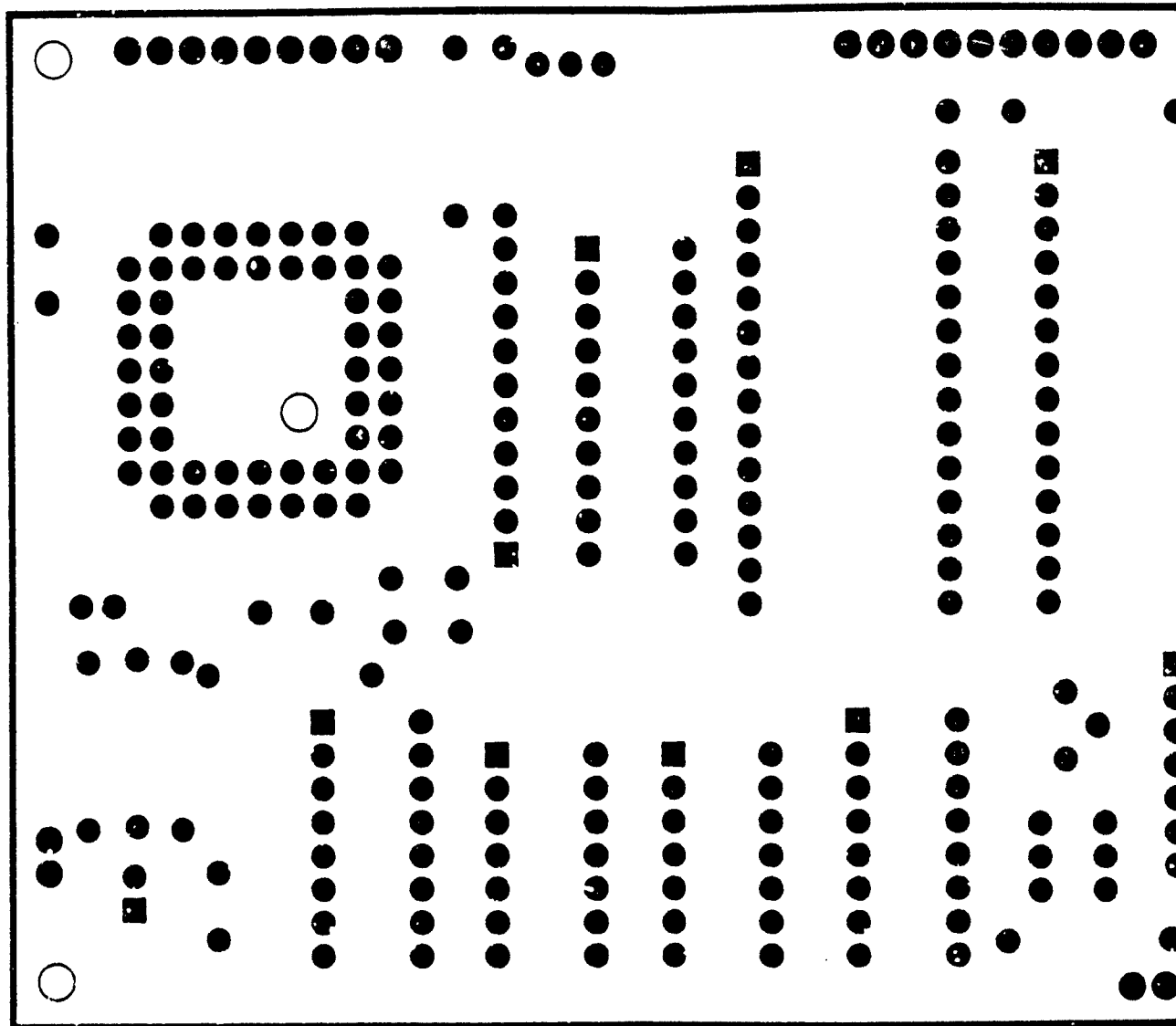
SIZE B	FSCM NO.	DRAWING NO. HHM-CPU-PWM
SCALE 2/1		SHEET 4 OF 6



4.0 INCHES

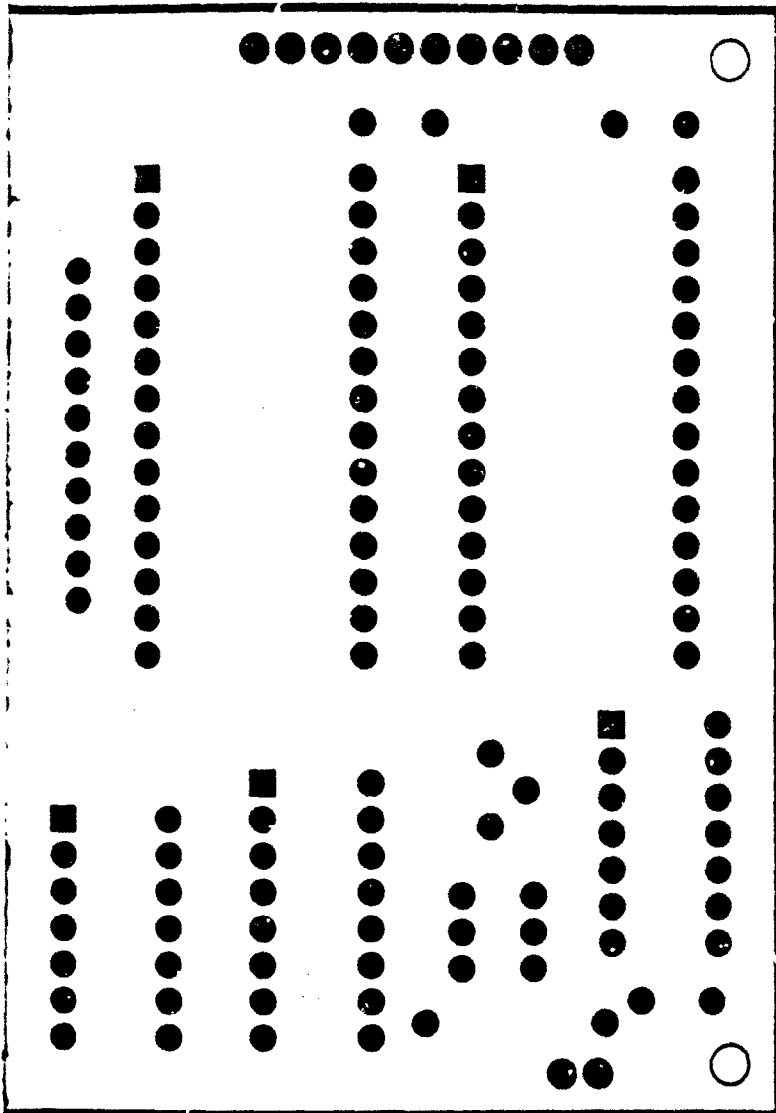


REDUCE TO THIS DIMENSION



INCHES

THIS DIMENSION



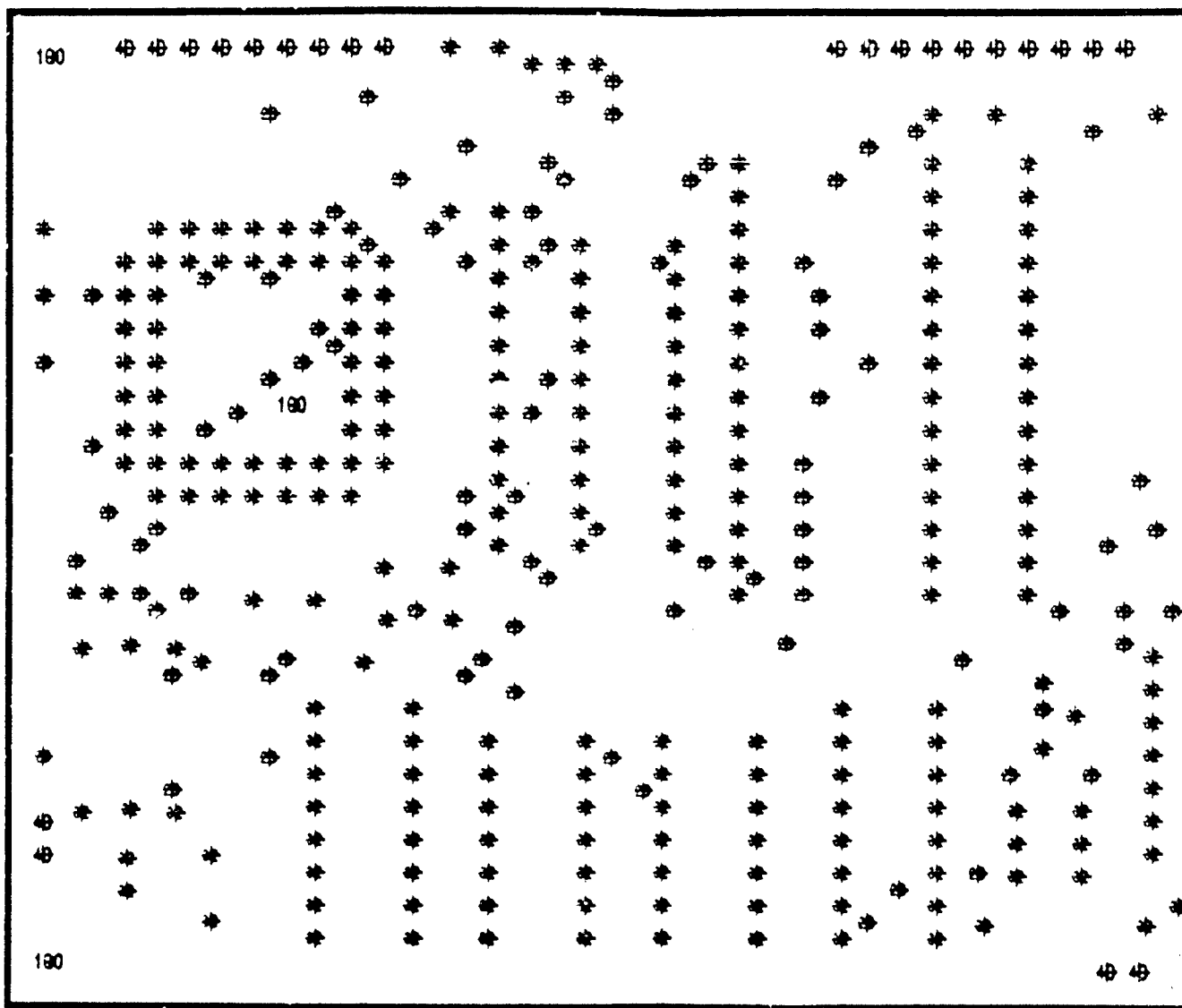
SOLDER MASK LAYER

SIZE B	F9CM NO.	DRAWING NO. HHM-CPU-PWM
SCALE 2/1		SHEET 5 OF 6



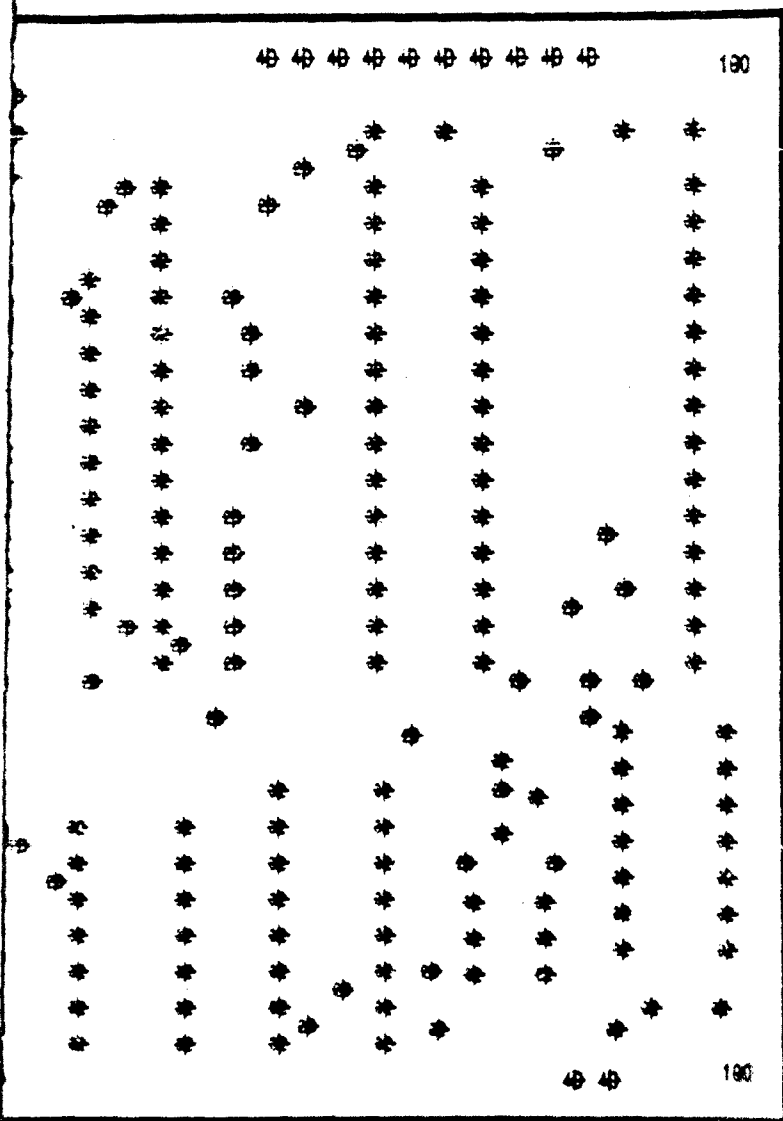
4.0 INCHES

REDUCE TO THIS DIMENSION



INCHES

THIS DIMENSION



DRILL MASTER LAYER

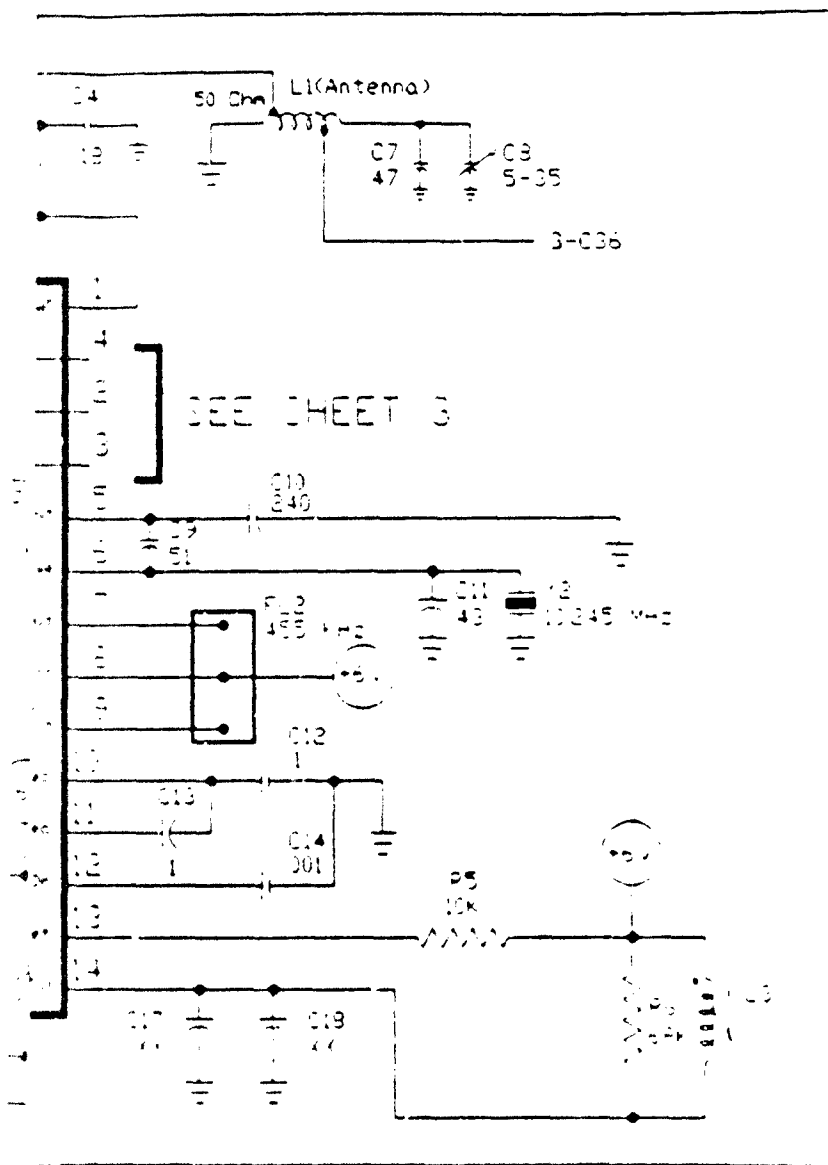
SIZE B	FSCM NO.	DRAWING NO. HHM-CPU-PWM
SCALE 2/1		SHEET 6 OF 6

PARTS W. A. HILLENBRAND BICMED. DAMD17-87-C-7195 HHM-CPU-PL 9/10/89
 LIST ENGR. CENTER, PURDUE UNIV.

HHM MAIN BOARD

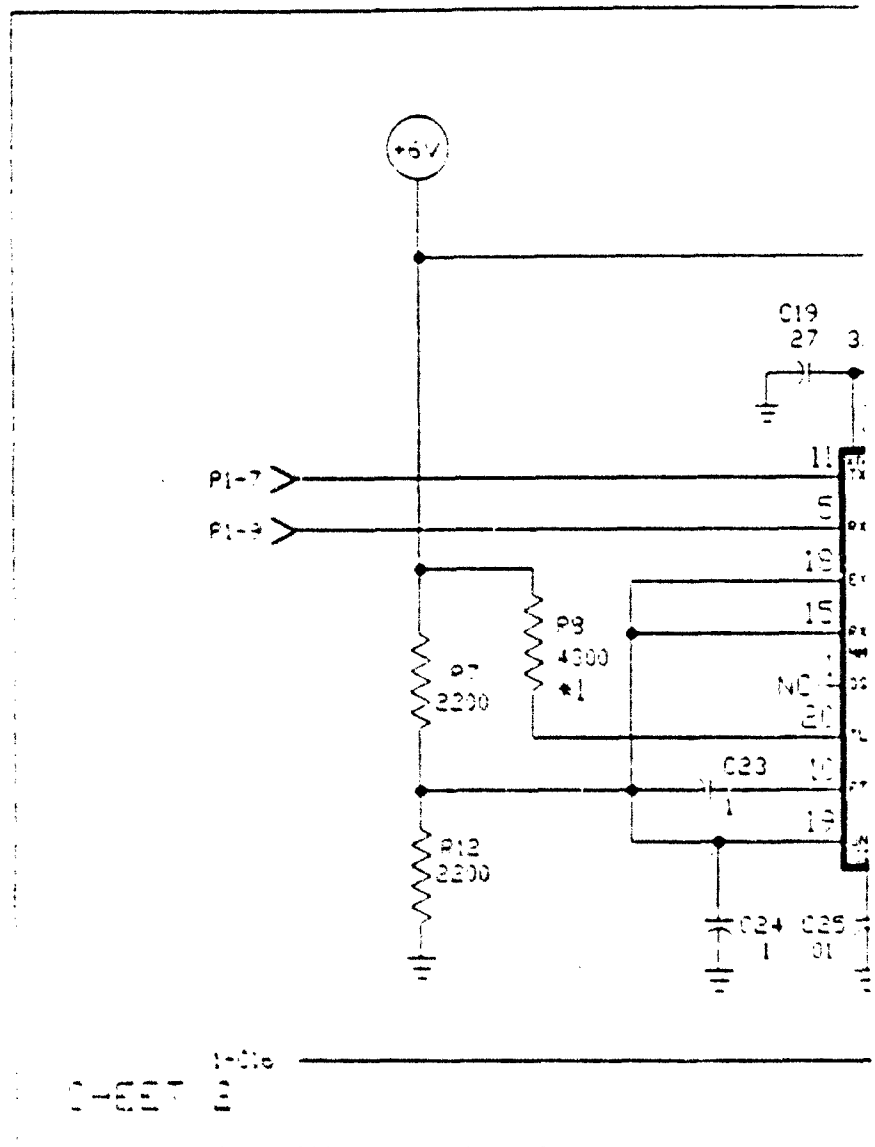
SHEET 1 OF 1

QTY	COMP-NAME	REFERENCE-DESIGNATOR	DESCRIPTION
2	RESV	R2 R3	VAL=100K
1	RESV	R6	VAL=62K
1	RESV	R4	VAL=4.7K
1	RESV	R1	VAL=6.8M
8	HC00	IC8 IC8 IC9 IC9 IC9 IC8 IC9 IC8	
1	AD589	IC6	
5	CAPV	C3 C8 C5 C6 C7	VAL=0.1UF
1	POLCAPV	C4	VAL=1UF
1	HC573R	IC2	
1	XTALV	CR1	VAL=8.0MHZ
7	RSIP10A	RN1 RN1 RN1 RN1 RN1 RN1 RN1	VAL=100K
2	RSIP10A	RN1 RN1	VAL=100K
2	CAPH	C1 C2	VAL=22PF
1	68HC11A1FN	IC1	
1	HC138	IC3	
1	8464	IC4	
1	27C64	IC5	
1	TPMPOTV	P7	VAL=100K
1	RESH	P5	VAL=100K
1	S8054HN	IC7	

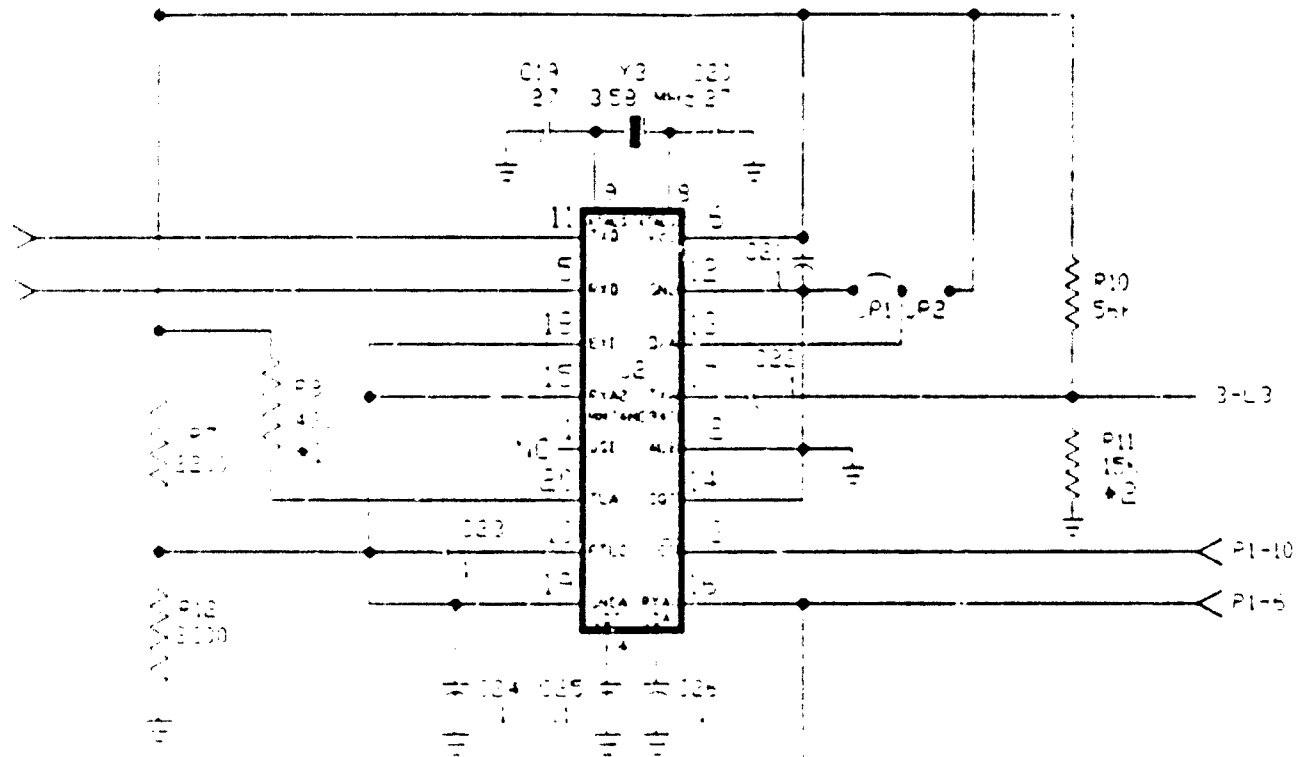


DOCUMENT SUPPLIED BY MAGNAVOX

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. ELLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE		
DRAWN GO	4-10-89		
CHECKED JTI	4-6-89	HHM RADIO COMMUNICATIONS LINK SCHEMATIC	
		SIZE B	DRAWING NO. HHM-RAD-SCH
		SCALE N.A	SHEET 1 OF 3



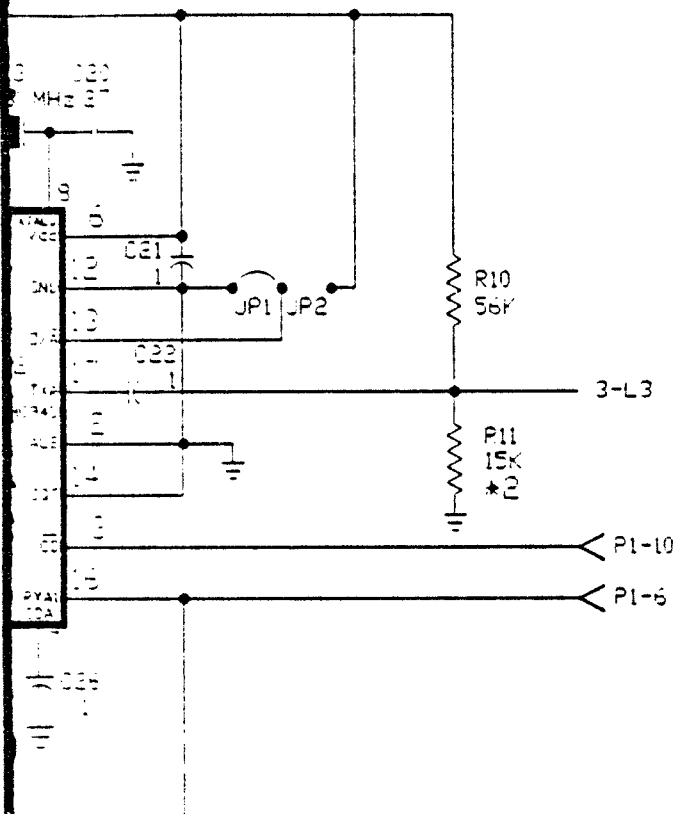
- *1 Select for 4kHz Deviation +/-200Hz
 - *2 Select for $F_c=72.170$ MHz +/-2kHz
- (NOMINAL VALUES SHOWN)



DI

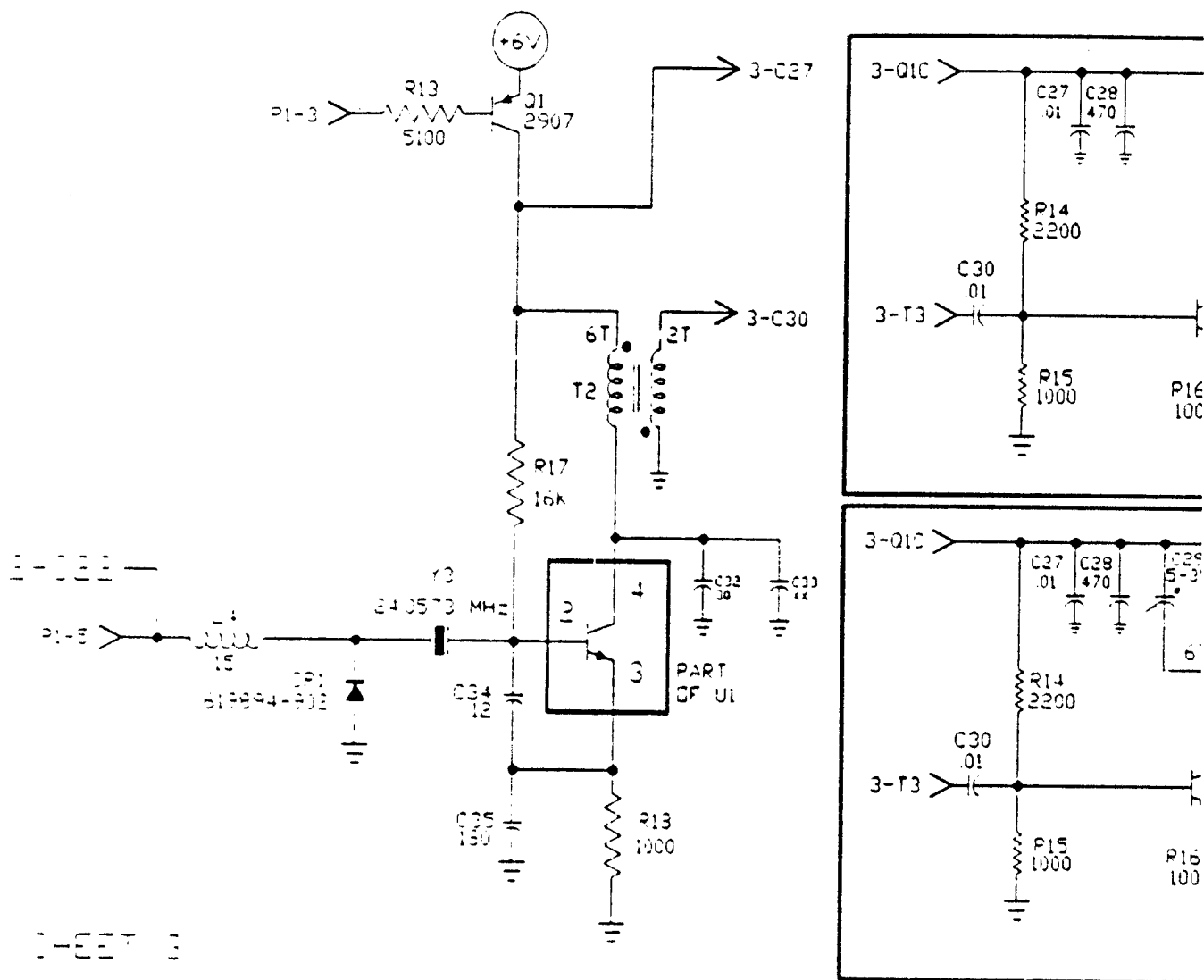
SIZE	FSCM
B	
SCALE	N/A

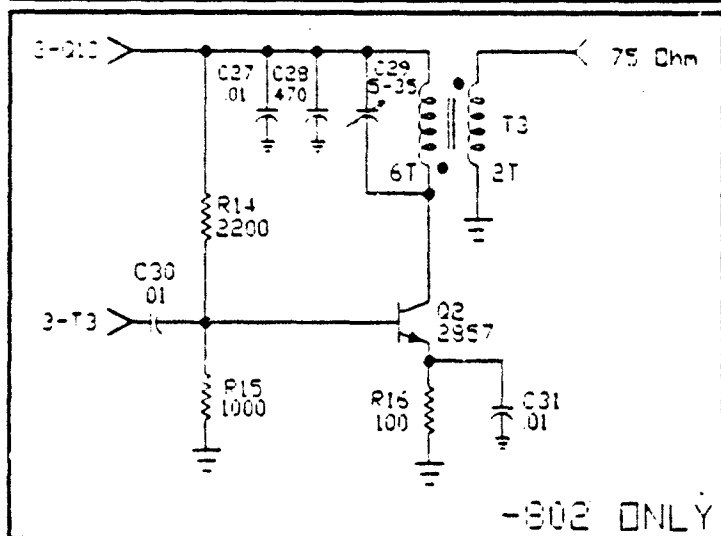
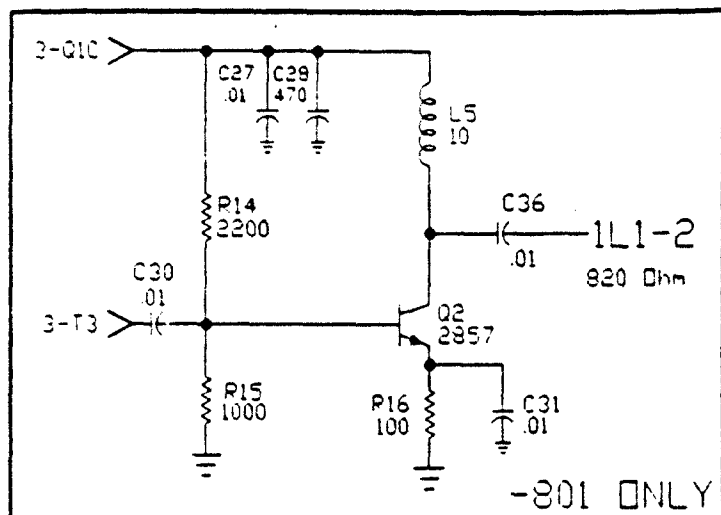
*1 Select for 4kHz Deviation +/-200Hz
*2 Select for Fc=72.170 MHz +/-2kHz
(NOMINAL VALUES SHOWN)



DOCUMENT SUPPLIED BY MAGNAVOX

SIZE B	FSCM NO.	DRAWING NO. HHM-RAD-SCH		
SCALE N/A			SHEET 2	OF 3

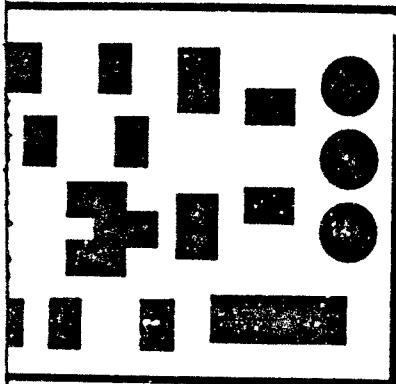




NOTES: 1. HHM RADIO LINK USES ASSY -802

DOCUMENT SUPPLIED BY MAGNAVOX

SIZE B	FSC M NO.	DRAWING NO. HHM-RAD-SCH
SCALE N/A		SHEET 3 OF 3

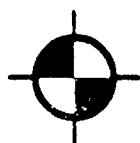
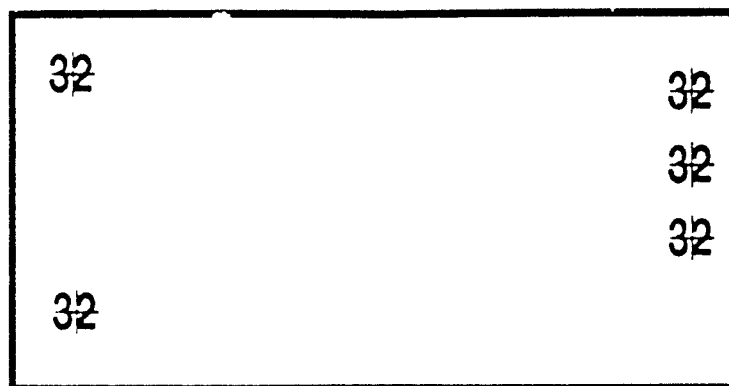


TO 1.0"



SOLDER MASK LAYER

SIZE B	FSCM NO.	DRAWING NO. PMC-PXR-PWM
SCALE 4/1		SHEET 4 OF 6



REDUCE TO 1.0"



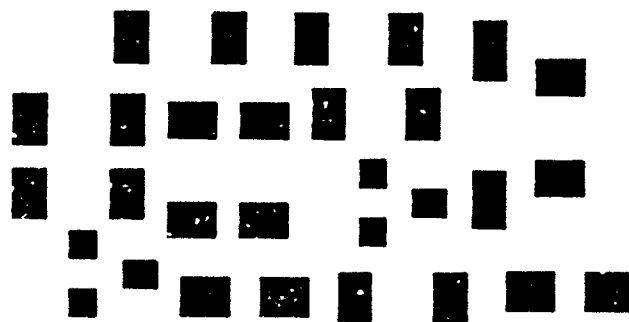
32
32
32

TO 1.0"



DRILL MASTER LAYER

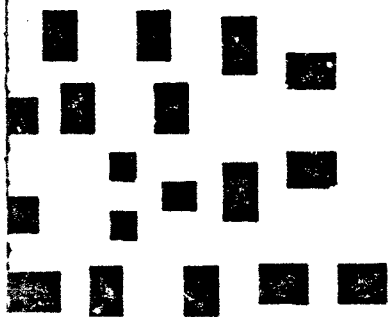
SIZE B	FSCM NO.	DRAWING NO. PMC-PXR-PWM
SCALE 4/1		SHEET 5 OF 6



I

I

SIZE
B
SCALE



SOLDER PASTE LAYER

SIZE B	FSCM NO.	DRAWING NO. PMC-PXR-PWM
SCALE 4/1		SHEET 6 OF 6

PARTS W. A. HILLENBRAND BIOMED. DAMD17-87-C-7195 PMC-PXR-PL 9/10/89
LIST ENGR. CENTER, PURDUE UNIV.

PMC PROXIMITY RECIEVER BOARD

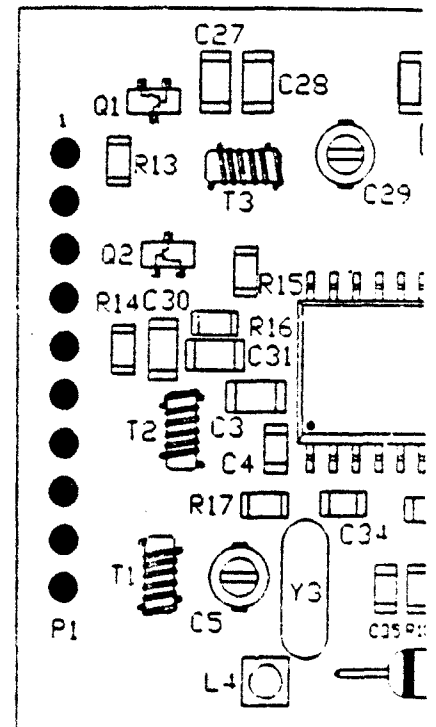
SHEET 1 OF 1

QTY	REFERENCE-DESIGNATOR	DESCRIPTION
2	R8 R3	VAL=10K
2	R7 R4	VAL=200
1	R2	VAL=68K
1	R1	VAL=360K
1	R9	VAL=390K
1	R5	VAL=56K
1	R6	VAL=430K
1	C4	VAL=500pF
1	C3	VAL=470pF
1	Q2	2N3906
1	Q1	2N3904
1	D1	1N4148

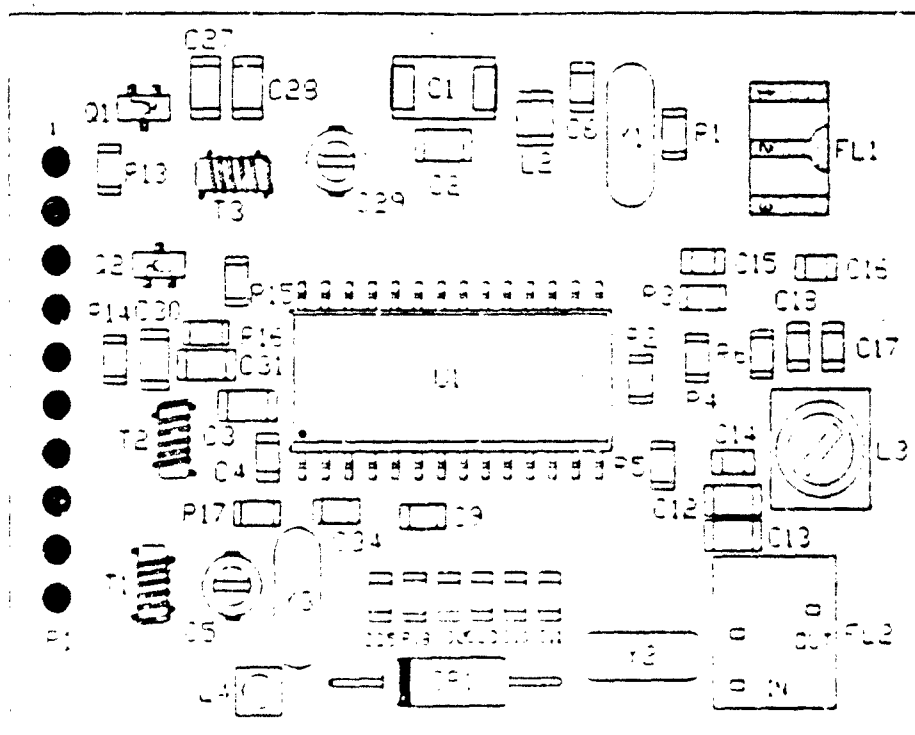
COMPONENT PLA

<-802

GROUND
+6VDC
KEY LINE
CPAPE
TX AUDIO
RX AUDIO
TX DATA
+6VDC
RX DATA
CARRIER DETECT



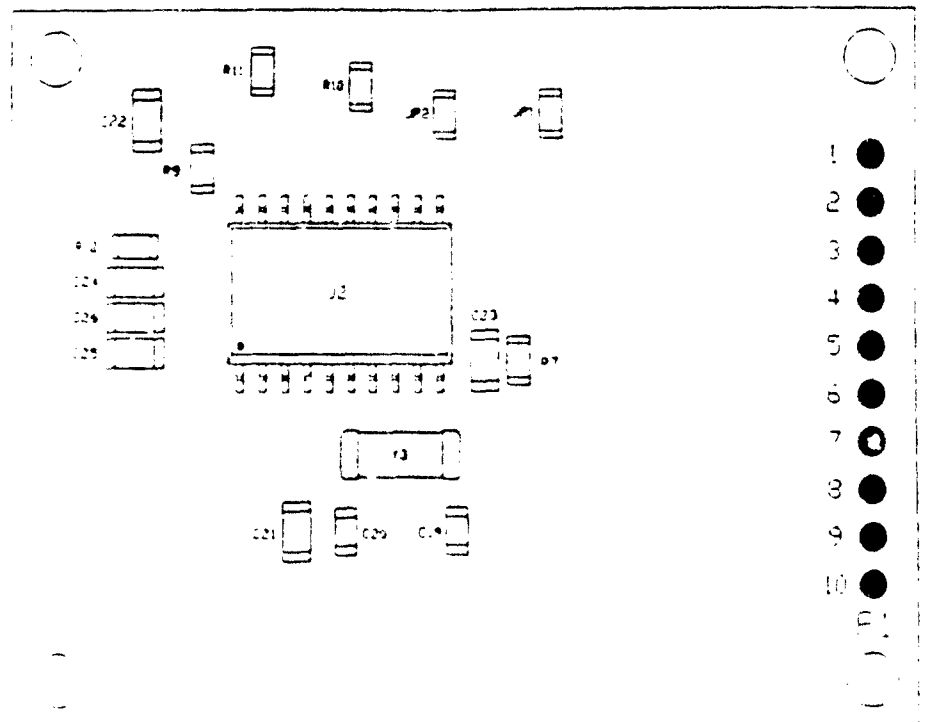
COMPONENT PLACEMENT - FRONT (-802 ONLY)



DOCUMENT SUPPLIED BY MAI

CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMET WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	HHM RADIO I PRINTED	
DRAWN GW	4-2-89		
CHECKED JTT	11-6-89	SCALE N/A	
		SIZE B	FORM NO.

COMPONENT PLACEMENT -



- | | |
|------|----------------|
| 1 ● | GROUND |
| 2 ● | +6VDC |
| 3 ● | KEY LINE |
| 4 ● | SPARE |
| 5 ● | TX AUDIO |
| 6 ● | RX AUDIO |
| 7 ● | TX DATA |
| 8 ● | +6VDC |
| 9 ● | RX DATA |
| 10 ● | CARRIER DETECT |

11-

SIZE B	FSCM NO.	DRAWING NO. HHM-RAD-PWM
SCALE N/A		SHEET 2 OF 2

HHM RADIO COMMUNICATIONS LINK BOARD

SHEET 1 OF 1

DISPATCH EXTERNAL PARTS LIST FOR ASSEMBLY ES-502-8772-02-01-501

REVISION PAGE 1

MODULE ASSEMBLY--PURDUE-PNC DIGITAL/AF

DRAWN REV 1 REV 1-A

PART NUMBER	ITEM	QTY	REF	REF	DESIGNATION	DESCRIPTION	NON-STD REF
0F045502	1	1	FL	2		FILTER, MULTI-ELEMENT LADDER	
0S000150M00	2	1	Y	4		RESONATOR, CERAMIC, CHIP MURATA	
ES-502-8772-02-01	3	1				PRINTED WIRING BOARD	
ES-502-8772-02-02	4	REF				DIAGRAM, SCHEMATIC	
SRM4217R101K50VPS	5	1	C	14		CAPACITOR, FIXED CERAMIC	
SRM4217R107K50VPS	6	5	C	15,25,27,32,33		CAPACITOR, FIXED CERAMIC	
SRM4217R108K50VPS	7	2	C	6,24		CAPACITOR, FIXED CERAMIC	
SRM4217R108K50VPS	8	1	C	4		CAPACITOR, FIXED CERAMIC	
SRM4217R108K50VPS	9	1	C	35		CAPACITOR, FIXED CERAMIC	
SRM4217R241K50VPS	10	1	C	10		CAPACITOR, FIXED CERAMIC	
SRM4217R270K50VPS	11	2	C	19,22		CAPACITOR, FIXED CERAMIC	
SRM4217R280K50VPS	12	1	C	32		CAPACITOR, FIXED CERAMIC	
SRM4217R400K50VPS	13	1	C	11		CAPACITOR, FIXED CERAMIC	
SRM4217R470K50VPS	14	1	C	28		CAPACITOR, FIXED CERAMIC	
SRM4217R470K50VPS	15	1	C	16		CAPACITOR, FIXED CERAMIC	
SRM4217R510K50VPS	16	1	C	9		CAPACITOR, FIXED CERAMIC	
SRM4217R510K50VPS	17	1	C	1,2,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100		CAPACITOR, FIXED CERAMIC, MURATA	
0T000107-02	18	1	C	2		INDUCTOR, VARIABLE, TUNING, BERNARD	
0T000107-02	19	1	C	1		MICROPROCESSOR, LINEAR, MOTOROLA	
0T000107-02	20	1	C	2		MICROPROCESSOR, DIGITAL, NATIONAL	
0T000107-02	21	1	C	2		TRANSISTOR, RF, SURFACE MOUNT	
0T000107-02	22	1	C	1		TRANSISTOR, SURFACE MOUNT	
0T000107-02	23	1	FL	1		FILTER, CERAMIC, CHIP MURATA	
0T000107-02	24	1	C	1		CAPACITOR, FIXED, TANTALUM	
0T000107-02	25	1	C	4		INDUCTOR, FIXED, CHIP AMERICAN ELECTRONIC INC.	
0T000107-02	26	1	C	2		INDUCTOR, FIXED, CHIP, COILCRAFT	
0T000107-02	27	1	R	16		RESISTOR, FIXED, FILM, CHIP, 100 ohm	
0T000107-02	28	1	R	1		RESISTOR, FIXED, FILM, CHIP, 100 ohm	
0T000107-02	29	2	R	15, 18		RESISTOR, FIXED, FILM, CHIP, 10 ohm	
0T000107-02	30	2	R	1, 12, 14		RESISTOR, FIXED, FILM, CHIP, 2.2k ohm	
0T000107-02	31	1	R	5		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	32	2	R	3, 10		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	33	1	R	11		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	34	1	R	21, 5		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	35	1	R	17		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	36	1	R	4		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	37	1	R	10		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	38	1	R	6		RESISTOR, FIXED, FILM, CHIP, 15.1k ohm	
0T000107-02	39	1	C	7		CAPACITOR, FIXED, CHIP	
0T000107-02	40	2	C	3, 24		CAPACITOR, VARIABLE, 0.05-0.01 ohm	
0T000107-02	41	1	C	1		SEMICONDUCTOR, DIODE, VARIACAP	
0T000107-02	42	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	43	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	44	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	45	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	46	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	47	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	48	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	49	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	
0T000107-02	50	1	C	1		CRYSTAL, QUARTZ, 10.7 MHz	

ARMY HAND HELD MONITOR A

COIL FORM

O.D. 3/8" dia. TYGON

O.D. 3

The Coil form consists of two pieces of tygon, the outer has an O.D. 3/8" dia.. The outer piece is shrunk on a smaller dia. of tygon tubing, this will give enough rigidity so that a coil can be wound on it, yet be flexible.

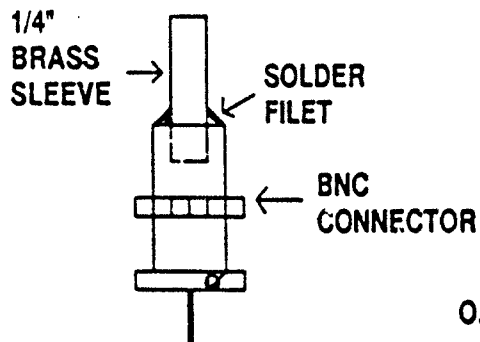
SECONDARY

25 turns
#20 AWG
spaced
over
form

25 turns
#20 AWG
Close
wound

1.5 turns
over primary

BASE ASSEMBLY



O.D. 1/4" dia. TYGON

BASE ASSEMBLY

ARMY HAND HELD MONITOR ANTENNA

72 MHZ

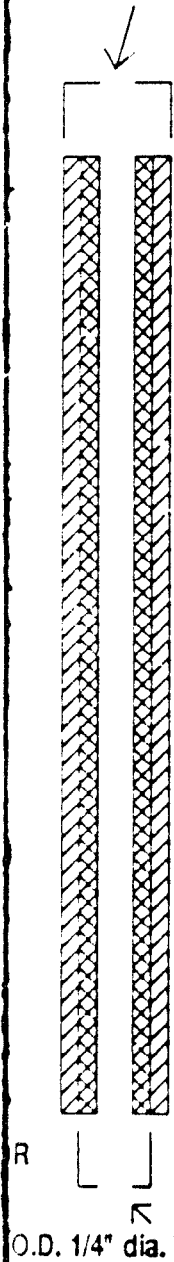
COIL FORM

MECHANICAL

ELECTRICAL SCHEMATIC

O.D. 3/8" dia. TYGON

O.D. 3/8" dia. TYGON



SECONDARY

25 turns
#20 AWG
spaced
over
form

25 turns
#20 AWG
Close
wound

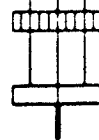
1.5 turns
over primary

BASE ASSEMBLY

COIL FORM

PRIMARY

11 turns
#20 AWG

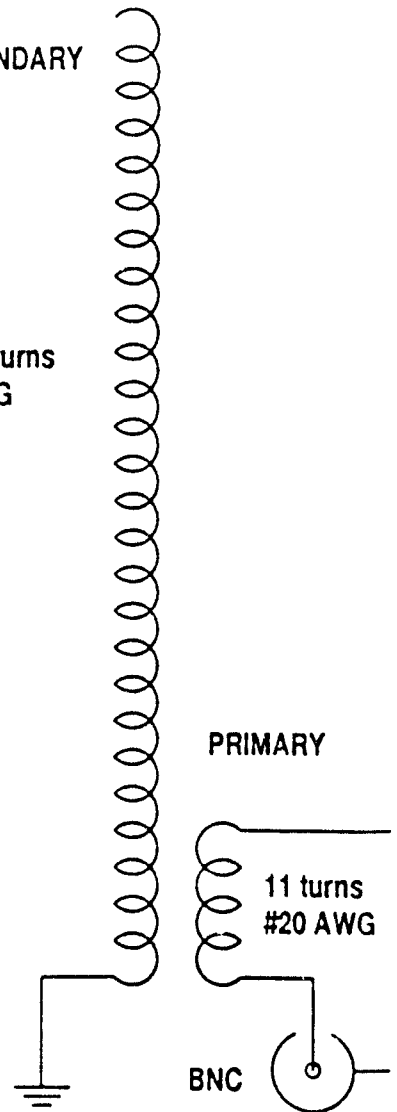


SECONDARY

51.5 turns
#20 AWG

PRIMARY

11 turns
#20 AWG



CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVER: W. A. HILLENBRA WEST LAFAYETT	
APPROVALS	DATE		
DRAWN DW	9-10-89		
CHECKED JFI	11-6-89		
		SIZE B	FSCM N
		SCALE	N/A

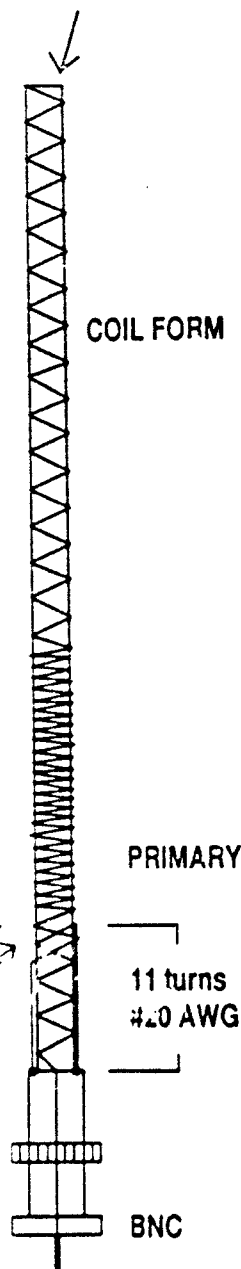
ANTENNA

72 MHZ

MECHANICAL

ELECTRICAL SCHEMATIC

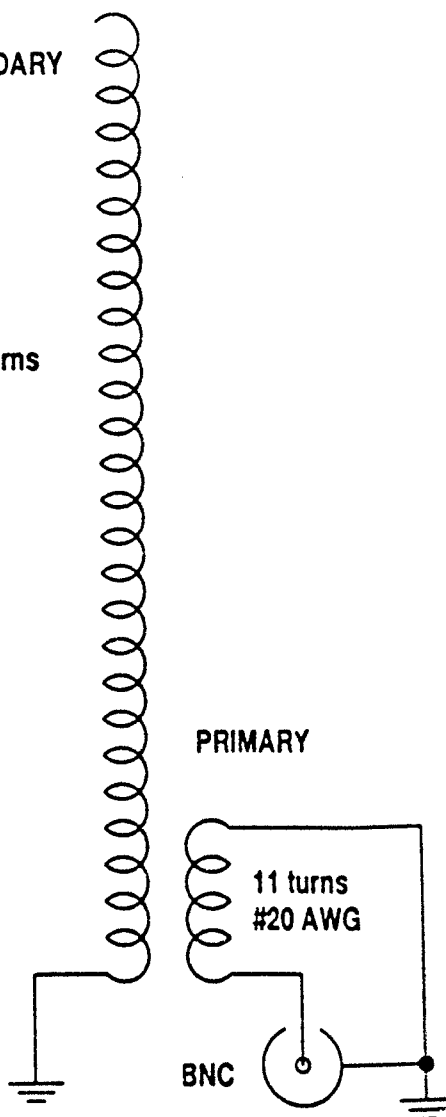
3/8" dia. TYGON



SECONDARY

51.5 turns
#20 AWG

PRIMARY



CONTRACT NO. DAMD17-87-C-7195		PURDUE UNIVERSITY W. A. HILLENBRAND BIOMEDICAL ENGINEERING CENTER WEST LAFAYETTE, IN 47907	
APPROVALS	DATE	HHM ANTENNA SCHEMATIC	
DRAWN DM	9-10-89		
CHECKED JFJ	11-6-89		
		SIZE B	FSCM NO.
		DRAWING NO. HHM-ANT-SCH	
		SCALE N/A	SHEET 1 OF 1

PROGRAMMING SOFTWARE
for the
MULTIMONITOR

mm.c

mm.c

```

/* */
/* Multiple Casualty Monitor */
/* Ver. 1.0 */
/* */
/* (c) 1989 Purdue University */
/* */

#include <stdio.h>
#include <hllreg.h>
#include <bmech.h>
#include <terminal.h>

/* display handler declarations */
#include "dispec.h"

#define LF      10
#define CR      13

#define PROXVAL 200
#define PROXMIN 5
#define MAXID   5
#define MAXRETRIES 3
#define EVALTIME 15

/* Received packet statuses */
#define GOODPAK 1
#define BADCHK  2
#define BADPAK  3
#define TIMEOUT -1

#define ACTIVE 1
#define INACTIVE 0

#define SHOW 1
#define NOSHOWN 0

#define MOTIONLT 1
#define MOTIONMOD 2
#define MOTICNHVY 3

/* structure used to set up serial interface */
struct ictrl {
    char      clk50; /* 50 HZ counter for real time clock */
    int       dcount; /* 50 HZ down counter for general purpose timing */
    int       stcount; /* 50 HZ down counter for status timing */
    int       scount; /* 50 HZ down counter for sleep routine */
    int       etime; /* 1 HZ evaluation time counter */

    char      tpakbuf[200]; /* buffer to construct packets for transmission */

    char      rpakbuf[50]; /* buffer to store received packets */
    char      *pakptr; /* pointer to receive packet buffer */
    int       rstate; /* state of receive packet finite state machine */

    int       chksum; /* check sum calculated on received packets */
    int       rchksum; /* check sum included in received packets */

    char      line[80]; /* buffer for display messages */

```

```

/* soldier status values returned from a PMC */
int      evnum;          /* evaluation number */
int      offw;           /* PMC off wrist flag */
int      hrate;          /* heart rate */
int      batt;           /* battery voltage */
int      mocount;        /* motion counter */
int      id;             /* PMC ID */

int      adflag;         /* enable A/D conversion flag */
int      adval;          /* value from A/D conversion */

int      retries;        /* number of retries for communication */

int      first;          /* flag to mark first evaluation cycle for a
                          * PMC */

char      '3;

char      clksec;
char      clkmin;
char      clkhr;

/* GG vars */
#define MAXCAS 20
#define WIN_SIZE 10
#define HR_HI 90
#define HR_LOW 50
#define HR_MIN 30

#define TIMEOUT 12
#define DTIMEOUT 3

char      addval;        /* ADD pushbutton current value */
char      deival;        /* DEL pushbutton current value */
char      upval;         /* UP pushbutton current value */
char      dval;          /* DWN pushbutton current value */
char      blpval;        /* HLP pushbutton current value */

char      addlast;       /* ADD pushbutton last value */
char      dellast;       /* DEL pushbutton last value */
char      uplast;        /* UP pushbutton last value */
char      dnlast;        /* DWN pushbutton last value */
char      blplast;       /* HLP pushbutton last value */

char      addstate;      /* ADD pushbutton state */
char      deistate;      /* DEL pushbutton state */
char      upstate;       /* UP pushbutton state */
char      dnstate;       /* DWN pushbutton state */
char      blpstate;      /* HLP pushbutton state */

char      addflag;       /* ADD pushbutton flag */
char      deiflag;       /* DEL pushbutton flag */
char      upflag;        /* UP pushbutton flag */
char      dnflag;        /* DWN pushbutton flag */
char      blpflag;       /* HLP pushbutton flag */

int      evalflag;       /* evaluate flag */

int      ncas;           /* current number of casualties */
int      curcas;         /* current polled casualty */
int      casptr;         /* casualty pointed to by cursor */

```

```

int          nextcasid;          /* next usable casid number */
struct cas {
    int      casid;
    int      pmcid;
    int      evnum, offw, hrate, batt, mocount;
    int      inhib;
    struct {
        unsigned hrhi:1, hrlo:1, lowpulse:1, offwrist:1, battlow:1, motion:2;
        unsigned pakerr:1, timeout:1;
        stat;
    }
} casualty[MAXCAS];

char "motion_str[]" = {
    "----",
    "NO",
    "YES",
    "YES",
    "YES"
};

char "stat_str[]" = {
    "----",
    "HRT RATE HIGH",
    "HRT RATE LOW",
    "LOW PULSE",
    "OFF WRIST",
    "BATTERY LOW",
    "COM ERROR",
    "COM LOST"
};

char null_hr_str[] = "???";

/* routine declarations */
void      int_setup();
__mod2__ void  OC1INT();
__mod2__ void  INTINT();

main()
{
    /* turn of transmitter off */
    key(0);
    /* initialize keyboard */
    HIIPACTL = 0x30 | HIIPACTL;
    HIIPORTA = 0xD0 | HIIPORTA;
    kbclr();
    /* initialize variables */
    casptr = 0;
    ncas = 0;
    nextcasid = 1;
    evalflag = 0;
    curcas = 0;
    /* set up interrupts */
    int_setup();
    /* set up serial port */
    tty_setup();
    /* initialize LCD display */
    disp_init();
    dcurs(0);
    dcurs(3);
    wmake(4, WIN_SIZE);

```

main

...main

```

/* Power up message */
help_screen();
pmc_screen();
pmc_show();
/* turn on A/D converter */
adon();
/* main processing loop */
while (1) {
    /* process any button pushes */
    if (addflag) {
        addcas();
        pmc_show();
        addflag = 0;
    } else if (deiflag) {
        delcas();
        pmc_show();
        deiflag = 0;
    } else if (hlpfld) {
        hlpflag = 0;
        help_screen();
        pmc_screen();
        pmc_show();
    } else if (arrows())
        pmc_show();
    check_cas(SHOW);
    if (adval >= 68 && stcount == 0) {
        sprintf(line, "Low Multimonitor Battery");
        dstatus(line);
    }
    stat_clr();
}

sleep(i)
    int i;
{
    scount = i * 50;
    while (scount);
}

prox()
{
    HIIPORTA |= 0x20;
    dcount = 5;
    while (dcount);
    HIIPORTA &= 0xdf;
}

dopss()
/* receive a PMC Soldier Status */
{
    int j;
    char *s;
    for (j = 0; rpakbuf[j] != '\0'; j++);
    if (strcmp(&rpakbuf[j + 1], "PSS", 3) != 0)
        return (0);
    j = j + 4;
    s = &rpakbuf[j];
    sscanf(s, "%d %d %d %d %d %d", &evnum, &offw, &brate, &mocount, &batt);
    casualty[curcas].evnum = evnum;
    casualty[curcas].offw = offw;
    casualty[curcas].brate = brate;
    casualty[curcas].mocount = mocount;
    casualty[curcas].batt = batt;
    casualty[curcas].stat.brhi = 0;
    casualty[curcas].stat.brlo = 0;
}

```

sleep

prox

dopss

mm.c

mm.c

...dopss

```

casualty[curcas].stat.lowpulse = 0;
casualty[curcas].stat.offwrist = 0;
casualty[curcas].stat.battlow = 0;
casualty[curcas].stat.motion = 1;
if (offw > 190) {
    casualty[curcas].stat.offwrist = 1;
    casualty[curcas].hrate = 0;
    casualty[curcas].mocount = 0;
} else {
    if (mocount >= 1 && mocount < 10) {
        casualty[curcas].stat.motion = 2;
    } else if (mocount >= 10 && mocount < 25) {
        casualty[curcas].stat.motion = 3;
    } else if (mocount >= 25) {
        casualty[curcas].stat.motion = 4;
    }
    if ((hrate == 0) || (hrate == -4)) {
        casualty[curcas].stat.lowpulse = 1;
    } else if ((hrate > 0) && (hrate <= HR_LOW)) {
        casualty[curcas].stat.hrlo = 1;
        alarm(2);
    } else if (hrate >= HR_HI) {
        casualty[curcas].stat.hrhi = 1;
        alarm(3);
    }
}
if (batt >= 68) {
    casualty[curcas].stat.battlow = 1;
}
return (1);
}

```

doppa()

doppa

```

{
    /* do Positive Packet Acknowledge */
    int j;
    char s;
    int lid;
    for (j = 0; rpakbuf[j] != '\0'; j++);
    if (strncmp(&rpakbuf[j + 1], "PPA", 3) != 0)
        return (0);
    j--;
    while (isdigit(rpakbuf[j] - 1))
        j--;
    s = &rpakbuf[j];
    sscanf(s, "%d", &lid);
    return (lid);
}

```

docss(d)

docss

```

{
    int id;
    /* send out Central Soldier Status query */
    char p;
    p = &rpakbuf[0];
    sprintf(p, "A%d CSS I", id);
    sendpak(p);
}

```

sendpak(pk)

sendpak

```

{
    char pk;
    int i;
    char p;
    p = pk;
    for (i = 0; p; p++)
        i = i + p;
}

```

...sendpak

```

i = i & 0xff;
*p = (i >> 4) + '0';
if (*p > '9')
    *p = *p + 7;
p++;
*p = (i & 0xf) + '0';
if (*p > '9')
    *p = *p + 7;
p++;
*p = LF;
p++;
*p = CR;
p++;
*p = 0;
key(1);
dcount = 10;
/* while(dcount, */
waitd();
p = pk;
pkputc('\r');
pkputc('X');
while (*pi) {
    pkputc(*pi);
    p++;
}
pkputc(CR);
pkputc(CR);
key(0);
}

pkputc(c)
    char    c;
{
    putchar(c);
}

getpacket(timeout, ptype, id)
    int      timeout;
    char     *ptype;
    int      id;
{
    int      done, c, j, lid;
    char     *p;
    rstate = 0;
    chksum = 0;
    pakptr = &rxbuf0;
    dcount = timeout * 50;
    done = 0;
    while (!) {
        while (com edb) done = 0;
        if (com edb) done = 0;
        return (TIMEOUT);
        c = getch() & 0xff;
        /* dputc(c); */
        /* detritus101; */
        switch (rstate) {
            case 0:
                if (c == '\r') {
                    *pakptr++ = c;
                    chksum += c;
                    rstate = 1;
                }
                break;
            case 1:
                if (c == '\n')

```

pkputc

getpacket

...getpacket

```

        break;
    if (c == 'A') {
        *pakptr++ = c;
        chksum += c;
        rstate = 2;
    } else
        pkabort();
    break;
case 2:
    if (c == LF || c == CR) {
        *pakptr = 0;
        s = pakptr;
        s--;
        s--;
        sscanf(s, "%x", &rchksum);
        chksum -= *s--;
        chksum -= *s;
        chksum = chksum & 0xff;
        *s-- = 0;
        *s = 0;
        if (chksum != rchksum)
            pkabort();
        else {
            for (j = 0; rpakbuf[j] != '\0'; j++);
            if (strcmp(&rpakbuf[j + 1], pktype, 3) == 0) {
                j--;
                while (isdigit(rpakbuf[j - 1]))
                    j--;
                s = &rpakbuf[j];
                sscanf(s, "%d", &lid);
                if ((lid == id) || (id == 0))
                    return (GOODPAK);
            }
        }
        /* not our packet, toss it away */
        pkabort();
    } else {
        *pakptr++ = c;
        chksum += c;
    }
    break;
}
if (pakptr >= &rpakbuf[49])
    pkabort();
}

```

pkabort()

pkabort

```

{
    pakptr = &rpakbuf[0];
    chksum = 0;
    rstate = 0;
}

```

```

mod2 void
INTINT()
{
}

```

INTINT

```

key(i)
int i;
{
    if (i == 0)
        HIIPORTA |= 0x08;
    else

```

key

...key

```

        H11PORTA &= 0xF7;
    }

    mod2 void
    OC1INT()
    {
        H11TFLG1 = 0x80;
        H11TOC1 -= 0x3c40;
        H11COPRST = 0x55;
        H11COPRST = 0xaa;
        /* check and debounce keys (keys are active low) */
        if ((H11PORTA & 0x04) == 0)
            b1pval = ACTIVE;
        else
            b1pval = INACTIVE;
        if (clk50 & 0x01) {
            if ((H11PORTA & 0x01) == 0)
                d1val = ACTIVE;
            else
                d1val = INACTIVE;
            if ((H11PORTA & 0x02) == 0)
                upval = ACTIVE;
            else
                upval = INACTIVE;
            /* set up for next read */
            H11PORTA = 0x40 | (H11PORTA & 0x3F);
        } else {
            if ((H11PORTA & 0x01) == 0)
                d1val = ACTIVE;
            else
                d1val = INACTIVE;
            if ((H11PORTA & 0x02) == 0)
                addval = ACTIVE;
            else
                addval = INACTIVE;
            /* set up for next read */
            H11PORTA = 0x80 | (H11PORTA & 0x3F);
        }
        if (b1pval + d1val + upval + d1val + addval > 1) {
            b1pval = INACTIVE;
            d1val = INACTIVE;
            upval = INACTIVE;
            d1val = INACTIVE;
            addval = INACTIVE;
        }
        dokey(&b1pval, &b1plast, &b1pstate, &b1pflag);
        dokey(&d1val, &d1plast, &d1state, &d1flag);
        dokey(&upval, &upplast, &upstate, &upflag);
        dokey(&d1val, &d1plast, &d1state, &d1flag);
        dokey(&addval, &addplast, &addstate, &addflag);
        if (addflag & clk50 == 0) {
            H11ADCTL = 0x10;
            while (H11ADCTL & 0x80 == 0);
            adval = H11ADR1;
        }
        if (dcount)
            dcount--;
        if (rcount)
            rcount--;
        if (wcount)
            wcount--;
        clk50++;
        if (clk50 >= 50) {
            clk50 = 0;

```

OC1INT

...OC1INT

```

        clk50 = 0;
        etime++;
    }
    if (clksec >= 60) {
        clkmin++;
        clksec = 0;
    }
    if (clkmin >= 60) {
        clkhr++;
        clkmin = 0;
    }
}

```

dokey(val, last, state, flag)

dokey

```

    char *val, *last, *state, *flag;
    {
        if (*state == INACTIVE) {
            if (*val == ACTIVE && *last == ACTIVE) {
                *state = ACTIVE;
                *flag = ACTIVE;
            }
        } else {
            if (*val == INACTIVE && *last == INACTIVE && *flag == INACTIVE) {
                *state = INACTIVE;
            }
        }
        *last = *val;
    }
}

```

```

mod2 void
COPINT()

```

COPINT

```

{
    dstatus0("COP RESET");
}

```

void

int_setup

```

int_setup()
{
    mod2 void OC1INT();
    mod2 void INTINT();
    mod2 void COPINT();
    void VECTOR();
    dcount = 0;
    scount = 0;
    stcount = 0;
    etime = 0;
    VECTOR(OC1INT, 9);
    VECTOR(INTINT, 14);
    VECTOR(COPINT, 17);
    VECTOR(COPINT, 18);
    HI1TOC1 = 0;
    HI1TMSK1 |= 0x80;
}

```

tty_setup()

tty_setup

```

{
    chksum = 0;
    rchksum = 0;
    rstate = 0;
    open();
    HI1BAUD = 0x35;
    ioctl(ttyb, GETPARAMS);
    ttyb->io_erase = CTRL('H');
    ttyb->io_kill = CTRL('U');
}

```

...tty_setup

```

    ttyb->io_flags = RAW;
    ioctl(ttyb, SETPARAMS);
}

```

com_rdy()

com_rdy

```

{
    if (H11SCSR & 0x20)
        return (1);
    else
        return (0);
}

```

adon()

adon

```

{
    H11OPTION |= 0x80;          /* power up a/d */
    H11PORTA |= 0x10;          /* power up wristz circuit */
    adflag = 1;
}

```

adoff()

adoff

```

{
    H11OPTION &= 0x7F;          /* power down a/d */
    H11PORTA &= 0xEF;          /* power up wristz circuit */
    adflag = 0;
}

```

```

/*****
/* */
/* GG routines
/* */
*****/

```

*/

help_screen()

help_screen

```

{
    ncls();
    nsetattr(DISP_NORM);
    nmessage(0, 0, " Multiple Casualty Monitor ");
    nmessage(1, 0, " ");
    nmessage(2, 0, " INSTRUCTIONS: ");
    nmessage(3, 0, " ");
    nsetattr(DISP_REV);
    nmessage(3, 0, "To Add Casualty To List.");
    nsetattr(DISP_NORM);
    nmessage(4, 0, " Put MM near PMC. Press <ADD>");
    nmessage(5, 0, " Wait for \"PMC ACTIVATED\" ");
    nmessage(6, 0, " Label body with number shown");
    nmessage(7, 0, " ");
    nsetattr(DISP_REV);
    nmessage(7, 0, "To Delete Casualty from List.");
    nsetattr(DISP_NORM);
    nmessage(8, 0, " Use arrow keys to select ");
    nmessage(9, 0, " casualty to be deleted ");
    nmessage(10, 0, " Press <DELETE>, the message ");
    nmessage(11, 0, " \"CONFIRM DELETION\" appears ");
    nmessage(12, 0, " Press <DELETE> again ");
    nmessage(13, 0, " ");
    nsetattr(DISP_REV);
    nmessage(13, 0, "To See This Screen.");
    nsetattr(DISP_NORM);
    nmessage(14, 0, " Press <HELP> ");
    nsetattr(DISP_RBLINK);
    nmessage(15, 0, "Press any key to exit HELP ");
    nsetattr(DISP_NORM);
    update();
    while ('keyhit())

```

...help_screen

```

        check_cas(NOSHOW);
        sleep(1);
        if (addflag) {
            kbclr();
            addflag = TRUE;
        } else
            kbclr();
        stcount = STIMEOUT * 50;
    }

```

pmc_screen()

pmc_screen

```

{
    nsetattr(DISP_REV);
    nmessage(0, 0, " Multiple Casualty Monitor  ");
    nsetattr(DISP_NORM);
    nmessage(1, 0, "CAUS HEART MOT.    STATUS    ");
    nmessage(2, 0, " NO. RATE                               ");
    nmessage(3, 0, "-----");
    nmessage(14, 0, "-----");
    nsetattr(DISP_REV);
    nmessage(15, 0, "Press <HELP> for instructions ");
    nsetattr(DISP_NORM);
    wsetattr(DISP_NORM);
    wcls();
}

```

pmc_show()

pmc_show

```

{
    int          i;
    int          n;
    int          status;
    char          hr_str[5];
    char          hr_sptr;

    /* alter window if necessary */
    if (n == offwin(casptr)) {
        if (n > 0)
            wscrollup(n);
        else
            wscrolldown(-n);
    }
    /* display casualties */
    domore();
    for (i = 0; i < ncas; i++) {
        wsettextpos(i, 0);
        if (status == maxstat(i)) {
            if (i == casptr)
                wsetattr(DISP_RBLINK);
            else
                wsetattr(DISP_BLINK);
        } else {
            if (i == casptr)
                wsetattr(DISP_REV);
            else
                wsetattr(DISP_NORM);
        }
        if (casualty[i].brate > HR_MIN) {
            sprintf(hr_str, "%3d", casualty[i].brate);
            hr_sptr = hr_str;
        } else {
            hr_sptr = null_hr_str;
        }
        printf(line, " %03d  %3s  %3s  %-14s", casualty[i].casid,
            hr_sptr, motion_str[casualty[i].stat.motion],
            stat_str[status]);
    }
}

```

...pmc_show

```

        wputs(line);
    }
    wupdate();
}

pmc_show1(i)
    int i;
{
    int status;
    char hr_str[5];
    char hr_sptr;

    /* alter window if necessary */
    /*
     * if(n = offwin(casptr)){ if(n > 0) wscrollup(n); else wscrolldown(-n);
     */
    /* update casualty window */
    wsettextpos(i, 0);
    if (status == maxstat(i)) {
        if (i == casptr)
            wsetattr(DISP_RBLINK);
        else
            wsetattr(DISP_BLINK);
    } else {
        if (i == casptr)
            wsetattr(DISP_REV);
        else
            wsetattr(DISP_NORM);
    }
    if (casualty[i].brate > 30) {
        sprintf(hr_str, "%3d", casualty[i].brate);
        hr_sptr = hr_str;
    } else {
        hr_sptr = null_hr_str;
    }
    sprintf(line, " %03d %3s %3s %-14s", casualty[i].casid,
        hr_sptr, motion_str[casualty[i].stat.motion],
        stat_str[status]);
    wputs(line);
    wupdl(i);
}

```

pmc_show1

```

dstatus(s)
    char *s;
{
    /* display status message (full line) */
    nsetattr(DISP_REV);
    nsettextpos(15, 0);
    nputs("");
    nsettextpos(15, 0);
    nputs(s);
    dupdln(15);
    stcount = TIMEOUT * 50;
}

```

dstatus

```

dstatus0(s)
    char *s;
{
    /* display primary status message (half line) */
    nsetattr(DISP_REV);
    nsettextpos(15, 0);
    nputs("");
    nsettextpos(15, 0);
    nputs(s);
    dupdln(15);
    stcount = TIMEOUT * 50;
}

```

dstatus0

...dstatus0

dstatus1

```

}

dstatus1(s)
    char *s;
    {
        /* display secondary status message (half
        * line) */
        nsetattr(DISP_REV);
        nsettextpos(15, 15);
        nputs(" ");
        nsettextpos(15, 15);
        nputs(s);
        dupdln(15);
        stcount = STIMEOUT * 50;
    }

stat_clr()
    {
        if (stcount == 0) {
            nsetattr(DISP_REV);
            nsettextpos(15, 0);
            nputs(" ");
            dupdln(15);
        }
    }

offwin(pos)
    int pos;
    {
        int i;

        i = wgetorg();
        if (pos < i)
            return (pos - i); /* scl dn neg */
        if (pos > WIN_SIZE + i - 1)
            return (pos - (WIN_SIZE + i - 1));
        return (0);
    }

inwin(pos)
    int pos;
    {
        int i;

        i = wgetorg();
        if (pos < i)
            return (-1);
        if (pos > WIN_SIZE + i - 1)
            return (-1);
        return (Win_row_dorg + pos - i);
    }

domore()
    {
        int i;

        i = wgetorg();
        nsetattr(DISP_NORM);
        if (i > 0) {
            nmessage(3, 0, "--More--");
            nputattr(3, 12, DISP_REV);
            nputattr(3, 13, DISP_REV);
            nputattr(3, 14, DISP_REV);
            nputattr(3, 15, DISP_REV);
        } else
    }

```

stat_clr

offwin

inwin

domore

...domore

```

        nmessage(3, 0, "-----");
    if (ncas > i + WIN_SIZE) {
        nmessage(14, 0, "v-----v-More-v-----v");
        nputatr(14, 12, DISP_REV);
        nputatr(14, 13, DISP_REV);
        nputatr(14, 14, DISP_REV);
        nputatr(14, 15, DISP_REV);
    } else
        nmessage(14, 0, "-----");
}

/* key board routines */
keyhit()
{
    if (hlpflag || dnflag || upflag || delflag || addflag)
        return (1);
    else
        return (0);
}

kbclr()
{
    addlast = INACTIVE;
    dellast = INACTIVE;
    uplast = INACTIVE;
    dnlast = INACTIVE;
    hlpast = INACTIVE;
    addflag = INACTIVE;
    delflag = INACTIVE;
    upflag = INACTIVE;
    dnflag = INACTIVE;
    hlpflag = INACTIVE;
    addstate = INACTIVE;
    deistate = INACTIVE;
    upstate = INACTIVE;
    dnstate = INACTIVE;
    hlpstate = INACTIVE;
}

arrows()
{
    if (upflag || dnflag) {
        if (upflag) {
            upflag = 0;
            if (casptr > 0)
                casptr--;
        } else if (dnflag) {
            dnflag = 0;
            if (casptr < ncas - 1)
                casptr++;
        }
        return (1);
    }
    return (0);
}

addcas()
{
    int          status;
    int          n;

    dstatus("Attempting PMC Activation");
    if (ncas >= MAXCAS) {
        dstatus("Limit Exceeded - PMC Not Added");
    }
}

```

keyhit

kbclr

arrows

addcas

mm.c

mm.c

...addcas

```

        sleep(1);
        return;
    }
    /* attempt PMC activation */
    retries = 0;
    do {
        /* pulse proximity transmitter */
        prox();
        /* check for PMC response */
        status = getpacket(2, "PPA", 0);
#ifdef BENCH
        dstatus(rpakbuf);
#endif
        if (status == GOODPAK)
            id = doppa();
        retries++;
        /* repeat activation attempt until successful or too */
        /* many retries */
    } while ((status != GOODPAK || id == 0) && retries < MAXRETRIES);
    if (retries >= MAXRETRIES)
        id = 0;
    addflag = 0;
    /* if id == 0 then activation was unsuccessful */
    if (id == 0) {
        dstatus("ERROR--Repeat Activate Steps");
        sleep(1);
        dstatus("Put MM Near PMC, Press <ADD>");
    } else if ((n = inlist(id)) != -1) {
        /* previously activated PMC */
        if (casualty[n].stat.timeout == 1) {
            dstatus("Casualty Reactivated");
            casualty[n].stat.timeout = 0;
            sleep(1);
        } else {
            dstatus("Casualty Already Active");
            sleep(1);
        }
    } else {
        casualty[ncas].pmcid = id;
        casualty[ncas].casid = nextcasid;
        casualty[ncas].evnum = 0;
        casualty[ncas].offw = 0;
        casualty[ncas].brate = 0;
        casualty[ncas].batt = 0;
        casualty[ncas].mocount = 0;
        casualty[ncas].inhib = 1;
        casualty[ncas].stat.brhi = 0;
        casualty[ncas].stat.brlo = 0;
        casualty[ncas].stat.lowpulse = 0;
        casualty[ncas].stat.offwrist = 0;
        casualty[ncas].stat.battlow = 0;
        casualty[ncas].stat.motion = 0;
        casualty[ncas].stat.pakerr = 0;
        casualty[ncas].stat.timeout = 0;
        casptr = ncas++;
        sprintf(line, "PMC %d ACTIVATED", id);
        dstatus(line);
        sleep(1);
        sprintf(line, "MARK CASUALTY WITH #%03d", nextcasid++);
        dstatus(line);
    }
}

delcas()
{

```

delcas

...delcas

```

if (ncas > 0) {
    /* confirm delete */
    dstatus("Hit <DELETE> to Confirm");
    deiflag = 0;
    dcount = DTIMEOUT * 50;
    while ('keyhit() && dcount);
    if (deiflag) {
        removecas();
        dstatus("");
    } else
        dstatus("Casualty Not Deleted");
} else
    dstatus("No Casualties to Delete");
}

removecas()
{
    int i, j;

    ncas--;
    for (i = casptr, i < ncas; i++) {
        j = i + 1;
        casualty[i].casid = casualty[j].casid;
        casualty[i].pmcid = casualty[j].pmcid;
        casualty[i].evnum = casualty[j].evnum;
        casualty[i].offw = casualty[j].offw;
        casualty[i].brate = casualty[j].brate;
        casualty[i].batt = casualty[j].batt;
        casualty[i].mocount = casualty[j].mocount;
        casualty[i].stat.hi = casualty[j].stat.hri;
        casualty[i].stat.w = casualty[j].stat.hri;
        casualty[i].stat.lo = casualty[j].stat.lowpulse;
        casualty[i].stat.offw = casualty[j].stat.offwrist;
        casualty[i].stat.batt = casualty[j].stat.battlow;
        casualty[i].stat.motio = casualty[j].stat.motion;
        casualty[i].stat.pakerr = casualty[j].stat.pakerr;
        casualty[i].stat.timeout = casualty[j].stat.timeout;
    }
    wsetattr(DISPLAY_NORM);
    wclr(ncas);
    if (casptr > ncas - 1)
        casptr = ncas - 1;
}

```

removecas

```

check_cas(showmode)
{
    int showmode;

    int cursor;

    if (etime >= EVALTIME) {
        evalflag = 1;
        etime = 0;
    }
    if (evalflag) {
        if (curcas < ncas) {
            if (casualty[curcas].inhib)
                casualty[curcas].inhib = 0;
            else
                getstatus(casualty[curcas].pmcid);
            if (showmode && ((cursor = inwin(curcas)) != -1)) {
                /* do cursor, get and show status */
                dsetcurpos(cursor, 0);
                dcursor(1);
                pmc_show(cursor);
            }
        }
    }
}

```

check_cas

mm.c

mm.c

...check_cas

```

        dcursor(0);
    }
    curcas++;
} else {
    curcas = 0;
    evalflag = 0;
}
}

getstatus(id)
    int          id;
{
    int          status;

    retries = 0;
    casualty[curcas].stat.pakerr = 0;
    casualty[curcas].stat.timeout = 0;
    do {
        docss(id);
        status = getpacket(2, "PSS", id);
        retries++;
    } while (status != GOODPAK && retries < MAXRETRIES);
    if (retries < MAXRETRIES)
        dopss();
    else {
        casualty[curcas].stat.timeout = 1;
    }
}

```

getstatus

```

inlist(id)
    int          id;
{
    int          i;
    /* return casid if pmcid is in casualty list */

    for (i = 0; i < ncas; i++) {
        if (casualty[i].pmcid == id)
            return (i);
    }
    return (-1);
}

```

inlist

```

alarm(nbeeps)
    int          nbeeps;
{
    while (nbeeps--) {
        dcount = 20;
        HIIPORTA = HIIPORTA & 0xef;
        waitd();
        HIIPORTA = HIIPORTA | 0x10;
        if (nbeeps) {
            dcount = 20;
            waitd();
        }
    }
}

```

alarm

```

maxstat(i)
    int          i;
{
    if (casualty[i].stat.timeout == 1)
        return (7);
    if (casualty[i].stat.pakerr == 1)
        return (6);
    if (casualty[i].stat.offwrist == 1)

```

maxstat

...mazstat

```

        return (4);
    if (casualty[i].stat.hrlow == 1)
        return (2);
    if (casualty[i].stat.hrhi == 1)
        return (1);
    if (casualty[i].stat.lowpulse == 1)
        return (3);
    if (casualty[i].stat.battlow == 1)
        return (5);
    return (0);
}

```

```

waitd()
{
    while (dcount)
        if (arrows())
            pmc_show();
}

```

waitd

```

checkd()
{
    if (dcount) {
        if (arrows())
            pmc_show();
        return (1);
    }
    return (0);
}

```

/* check dcount to see if still active */

checkd

displib.c

displib.c

```

/*****
/* dcln.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dcln(line)
{
    int line;

    int i;

    dsettextpos(line, 0);

    for (i = 0; i < DISP_COLS; i++)
        dputc(' ');

    dsettextpos(0, 0);
}

/*****
/* dcls.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dcls()
{
    int i, j;

    for (i = 0; i < DISP_ROWS; i++) {
        dsettextpos(i, 0);

        for (j = 0; j < DISP_COLS; j++)
            dputc(' ');

        dsettextpos(0, 0);
    }
}

/*****
/* dctrl.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dctrl(v)
{
    int v;

    /* send control byte to display */

    DISPAWAIT,

```

dcln

dcls

dctrl

...dctrl

```

        DISPCTL = v;
    }
    /* ..... */
    /* dctrl1.c: */
    /* AND1091 LCD display routine for use with the Multimonitor: */
    /* ..... */

#include <display.h>
dctrl1(v, data)
    int v, data;
{
    DISPWAIT, /* send display one data byte */
    DISPWAIT, /* and one control byte */
    DISPDATA = data;
    DISPWAIT,
    DISPCTL = v;
}
/* ..... */
/* dctrl2.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */

#include <display.h>
dctrl2(v, data1, data2)
    int v, data1, data2;
{
    DISPWAIT, /* send display two data bytes */
    DISPWAIT, /* and one control byte */
    DISPDATA = data1,
    DISPWAIT,
    DISPDATA = data2,
    DISPWAIT,
    DISPCTL = v;
}
/* ..... */
/* dcursor.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */

#include <display.h>
dcursor(mode)
    int mode;
{

```

dctrl1

dctrl2

dcursor

displib.c

displib.c

...dcursor

```

    if (mode)
        dctrl(0x9E);          /* DISPLAY CURSOR ON */

    else
        dctrl(0x9C);          /* DISPLAY CURSOR OFF */
}

/*****
 * dcursor.c:
 * AND1091 LCD display routine for use with the Multimonitor.
 *****/

#include <display.h>
dcursor(mode)
    int mode;

{

    if (mode)
        dctrl(0x9E);          /* DISPLAY CURSOR ON */

    else
        dctrl(0x9C);          /* DISPLAY CURSOR OFF */
}

/*****
 * dcurssize.c
 * AND1091 LCD display routine for use with the Multimonitor.
 *****/

#include <display.h>
dcurssize(size)
    int size;

{

    size = 0xA0 | (size & 0x07);

    dctrl(size),              /* CURSOR TYPE */
}

/*****
 * ddata.c
 * AND1091 LCD display routine for use with the Multimonitor.
 *****/

#include <display.h>
ddata(c)
    int c;

```

dcursor

dcurssize

ddata

...ddata.

```

{
    /* send data byte to display */

    DISPWAIT;

    DISPDATA = c;
}

/* disp_init.c */
/* AND1091 LCD display routine for use with the Multimonitors */
/* ..... */

#include <display.h>
disp_init()
{
    int i, j;

    while ((DISPCTL & 0x20) == 0),

    dctrl(0x84);          /* MODE SET */
    dctrl(0x40, 0, 0x0A); /* TXT HOME COM */
    dctrl(0x41, 0x1e, 0); /* TXT AREA COM */
    dctrl(0x42, 0, 0x08); /* GR HOME COM */
    dctrl(0x43, 0x1e, 0); /* GR AREA COM */
    dctrl(0x24, 0, 0x0A); /* ADDR PTR SET */
    dctrl(0x24, 0, 0x08); /* ADDR PTR SET */
    dctrl(0xA0);          /* CURSOR TYPE */
    dctrl(0x9C);          /* DISPLAY MODE SET */

    Disp_text_row = 0;
    Disp_text_col = 0;
    Disp_curs_row = 0;
    Disp_curs_col = 0;
    Disp_text_attr = DISP_NORM;
    Nxt_text_row = 0;
    Nxt_text_col = 0;
    Nxt_curs_row = 0;
    Nxt_curs_col = 0;
    Nxt_text_attr = DISP_NORM;

    for (i = 0; i < DISP_ROWS; i++) {
        for (j = 0; j < DISP_COLS; j++) {

```

disp_init

...disp_init

```

        Cur_img_ch[i][j] = 0xFF;
        Cur_img_at[i][j] = 0xFF;
    }
    ncis();
    dupdate();
}

/*****
/* dmessage.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dmessage(r, c, s)
    int      r, c;
    char     *s;
{
    dsettextpos(r, c);
    dputs(s);
}

/*****
/* dputattr.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dputattr(r, c, attr)
    int      r, c, attr;
{
    dsetattrpos(r, c);
    detrl1(0XC0, attr);
}

/*****
/* dputc.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dputc(c)
    int      c;

```

dmessage

dputattr

dputc

...dputc

```

        if (c == '\n') {
            Disp_text_col = 0;
            Disp_text_row = (Disp_text_row + 1) % DISP_ROWS;
        }
        else {
            c = c - 0x20;

            /* write char */
            dsettextpos(Disp_text_row, Disp_text_col);
            dctrl(0xc0, c);

            /* write char's attribute */
            dsetattrpos(Disp_text_row, Disp_text_col);
            dctrl(0xc0, Disp_text_attr);

            /* update row and col variables */
            Disp_text_col++;

            if (Disp_text_col >= DISP_COLS) {
                Disp_text_col = 0;
                Disp_text_row++;
                if (Disp_text_row >= DISP_ROWS) {
                    Disp_text_row = 0;
                }
            }
        }
    }
}
/...../

/* dputchar.c: */
/* AND1091 LCD display routine for use with the Multimonitor. */
/...../

#include <display.h>
dputchar(r, c, ch)
    int      r, c, ch;
{
    ch = ch - 0x20;
    dsettextpos(r, c);
    dctrl(0xc0, ch);
}
/...../

/* dputs.c: */

```

dputchar

```

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

```

```

#include <display.h>

```

```

dputs(s)
char *s;
{

```

dputs

```

    while (*s) {
        dputc(*s);
        s++;
    }
}
/*****

```

```

/* drc2aaddr.c:

```

*/

```

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

```

```

#include <display.h>

```

```

drc2aaddr(r, c, msb_ptr, lsb_ptr)
int r, c, *msb_ptr, *lsb_ptr;
{

```

drc2aaddr

```

    if (r < 8) {
        *lsb_ptr = r * 30 + c;
        *msb_ptr = DISP_AADDR1M;
    }

```

```

    else {
        *lsb_ptr = (r - 8) * 30 + c;
        *msb_ptr = DISP_AADDR2M;
    }
}
/*****

```

```

/* drc2taddr.c

```

*/

```

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

```

```

#include <display.h>

```

```

drc2taddr(r, c, msb_ptr, lsb_ptr)
int r, c, *msb_ptr, *lsb_ptr;
{

```

drc2taddr

```

    if (r < 8) {
        *lsb_ptr = r * 30 + c;
        *msb_ptr = DISP_TADDR1M;
    }
}

```

...drc2taddr

```

    }
    else {
        *lsb_ptr = (r - 8) * 30 + c;
        *msb_ptr = DISP_TADDR2M;
    }
}
/*****
/* dsetattrpos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dsetattrpos(r, c)
    int          r, c;
{
    int          lsb, msb;

    drc2aaddr(r, c, &msb, &lsb);
    dctrl2(0X24, lsb, msb);

    Disp_text_row = r;
    Disp_text_col = c;
}
/*****
/* dsetattr.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dsetattr(attr)
    int          attr;
{
    Disp_text_attr = attr;
}
/*****
/* dsetcurspos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dsetcurspos(r, c)
    int          r, c;
{

```

*/

dsetattrpos

*/

dsetattr

*/

dsetcurspos

...dsetcurspos

```

        if (r >= 8)
            r = r + 0X10 - 0X08;

        dctrl2(0X21, c, r);

        Disp_curs_row = r;

        Disp_curs_col = c;
    }
    /*.....*/
    /* dsettextpos.c */
    /* AND1091 LCD display routine for use with the Multimonitor: */
    /*.....*/

#include <display.h>
dsettextpos(r, c)
    int          r, c;
{
    int          lsb, msb;

    drc2taddr(r, c, &msb, &lsb);

    dctrl2(0X24, lsb, msb);

    Disp_text_row = r;

    Disp_text_col = c;
}
/*.....*/
/* dupdate.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/*.....*/

#include <display.h>
dupdate()
{
    int          i, j;

    for (i = 0; i < DISP_ROWS; i++) {
        for (j = 0; j < DISP_COLS; j++) {
            if (Cur_img_ch[i][j] != Nxt_img_ch[i][j]) {
                dsettextpos(i, j);

                dputchar(i, j, Nxt_img_ch[i][j]);

                Cur_img_ch[i][j] = Nxt_img_ch[i][j];
            }
            if (Cur_img_at[i][j] != Nxt_img_at[i][j]) {

```

dsettextpos

dupdate

...dupdate

```

        dsetattrpos(i, j);
        dputattr(i, j, Nxt_img_at[i][j]);
        Cur_img_at[i][j] = Nxt_img_at[i][j];
    }
}
Disp_text_row = Nxt_text_row;
Disp_text_col = Nxt_text_col;
Disp_curs_row = Nxt_curs_row;
Disp_curs_col = Nxt_curs_col;
Disp_text_attr = Nxt_text_attr;
dsetcurspos(Nxt_curs_row, Nxt_curs_col);
}
/*****
/* dupdln.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
dupdln(i)
    int i;
{
    int j;

    for (j = 0; j < DISP_COLS; j++) {
        if (Cur_img_ch[i][j] != Nxt_img_ch[i][j]) {
            dsettextpos(i, j);
            dputchar(i, j, Nxt_img_ch[i][j]);
            Cur_img_ch[i][j] = Nxt_img_ch[i][j];
        }
        if (Cur_img_at[i][j] != Nxt_img_at[i][j]) {
            dsetattrpos(i, j);
            dputattr(i, j, Nxt_img_at[i][j]);
            Cur_img_at[i][j] = Nxt_img_at[i][j];
        }
    }
    Disp_text_row = Nxt_text_row;
    Disp_text_col = Nxt_text_col;
    Disp_curs_row = Nxt_curs_row;

```

dupdln

...dupdln

```

    Disp_curs_col = Nxt_curs_col;
    Disp_text_attr = Nxt_text_attr;
    dsetcurspos(Nxt_curs_row, Nxt_curs_col);
}
/*****
/* ncln.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
ncln(line)
{
    int line;

    int i;

    nsettextpos(line, 0);
    for (i = 0; i < DISP_COLS; i++)
        nputc(' ');
    nsettextpos(0, 0);
}
/*****
/* ncls.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
ncls()
{
    int i, j;

    for (i = 0; i < DISP_ROWS; i++) {
        nsettextpos(i, 0);
        for (j = 0; j < DISP_COLS; j++)
            nputc(' ');
    }
    nsettextpos(0, 0);
}
/*****
/* nmessage.c:

```

ncln

ncls

```

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

```

```

#include <display.h>

```

```

nmessage(r, c, s)

```

nmessage

```

    int      r, c;
    char     *s;
{

```

```

    nsettextpos(r, c);

```

```

    nputs(s);

```

```

}

```

```

/*****

```

```

/* nputattr.c

```

*/

```

/* AND1091 LCD display routine for use with the Multimonitor: */

```

```

/*****

```

```

#include <display.h>

```

```

nputattr(r, c, attr)

```

nputattr

```

    int      r, c, attr;
{

```

```

    Nxt_img_at[r][c] = attr;

```

```

}

```

```

/*****

```

```

/* nputc.c

```

*/

```

/* AND1091 LCD display routine for use with the Multimonitor: */

```

```

/*****

```

```

#include <display.h>

```

```

nputc(c)

```

nputc

```

    int      c;
{

```

```

    if (c == '\n') {

```

```

        Nxt_text_col = 0;

```

```

        Nxt_text_row = (Nxt_text_row + 1) % DISP_ROWS;

```

```

    }

```

```

    else {

```

```

        /* write char */

```

```

        nsettextpos(Nxt_text_row, Nxt_text_col);

```

```

        /* write char's attribute */

```

```

        nsetattrpos(Nxt_text_row, Nxt_text_col);

```

```

        Nxt_img_ch[Nxt_text_row][Nxt_text_col] = c;

```

```

        Nxt_img_at[Nxt_text_row][Nxt_text_col] = Nxt_text_attr;

```



```

/* update row and col variables */
Nxt_text_col++;

if (Nxt_text_col >= DISP_COLS) {
    Nxt_text_col = 0;
    Nxt_text_row++;
    if (Nxt_text_row >= DISP_ROWS) {
        Nxt_text_row = 0;
    }
}

}

/* nputchar.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */

#include <display.h>
nputchar(r, c, ch)
    int      r, c, ch;
{
    Nxt_img_ch[r][c] = ch;
}

/* nputs.c */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */

#include <display.h>
nputs(s)
    char      *s;
{
    while (*s) {
        nputc(*s);
        s++;
    }
}

/* nsetattrpos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/* ..... */

#include <display.h>
nsetattrpos(r, c)
    int      r, c;

```

...nputc

*/

nputchar

*/

nputs

*/

nsetattrpos

displib.c

displib.c

...nsetattrpos

```

{
    Nxt_text_row = r;
    Nxt_text_col = c;
}
/...../
/* nsetattrpos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
nsetattr(attr)
    int          attr;
{
    Nxt_text_attr = attr;
}
/...../
/* nsetcurspos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
nsetcurspos(r, c)
    int          r, c;
{
    Nxt_curs_row = r;
    Nxt_curs_col = c;
}
/...../
/* nsettextpos.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/...../

#include <display.h>
nsettextpos(r, c)
    int          r, c;
{
    Nxt_text_row = r;
    Nxt_text_col = c;
}
/...../

```

nsetattr

nsetcurspos

nsettextpos

displib.c

displib.c

```

/* wcln.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****/

#include <display.h>
wcln(line)
    int line;
{
    int i;

    wsettextpos(line, 0);
    for (i = 0; i < DISP_COLS; i++)
        wputc(' ');
    wsettextpos(0, 0);
}
/*****/

/* wcls.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****/

#include <display.h>
wcls()
{
    int i, j;

    for (i = 0; i < WIN_ROWS; i++) {
        wsettextpos(i, 0);
        for (j = 0; j < DISP_COLS; j++)
            wputc(' ');
    }
    wsettextpos(0, 0);
}
/*****/

/* wgetorg.c: */
/* AND1091 LCD display routine for use with the Multimonitor: */
/*****/

#include <display.h>
wgetorg()
{
    return (Win_row_org);
}

```

wcln

wcls

wgetorg

```

/*****
 * win_to_nxt.c:
 * AND1091 LCD display routine for use with the Multimonitor:
 *****/

#include <display.h>
win_to_nxt()
{
    int i, j, nrow, wrow;

    nrow = Win_row_dorg;
    wrow = Win_row_org;
    for (i = 0; i < Win_row_len; i++, nrow++, wrow++) {
        for (j = 0; j < DISP_COLS; j++) {
            nputchar(nrow, j, Win_ch[wrow][j]);
            nputattr(nrow, j, Win_at[wrow][j]);
        }
        nsettextpos(0, 0);
    }
}

/*****
 * win_to_nln.c:
 * AND1091 LCD display routine for use with the Multimonitor:
 *****/

#include <display.h>
win_to_nln(wrow)
{
    int j, nrow;
    int offset;

    offset = wrow - Win_row_org;
    if (offset >= Win_row_len || offset < 0)
        return;
    nrow = Win_row_dorg + offset;
    for (j = 0; j < DISP_COLS; j++) {
        nputchar(nrow, j, Win_ch[wrow][j]);
        nputattr(nrow, j, Win_at[wrow][j]);
    }
}

```

win_to_nxt

win_to_nln

displib.c

displib.c

```

/*****
/* wmake.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

*/

```

#include <display.h>
wmake(drow, len)
    int drow, len;
{
    Win_row_len = len;
    Win_row_org = 0;
    Win_row_dorg = drow;
    wsetattr(DISP_NORM);
}

```

wmake

```

/*****
/* wmessage.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

*/

```

#include <display.h>
wmessage(r, c, s)
    int r, c;
    char *s;
{
    wsettextpos(r, c);
    wputs(s);
}

```

wmessage

```

/*****
/* wputattr.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

*/

```

#include <display.h>
wputattr(r, c, attr)
    int r, c, attr;
{
    Win_attr[r][c] = attr;
}

```

wputattr

```

/*****
/* wputc.c:
*****/

```

*/

```

/* AND1091 LCD display routine for use with the Multimonitor: */
/*****

#include <display.h>
wputc(c)
    int      c;
{

    if (c == '\n') {
        Win_text_col = 0;
        Win_text_row = (Win_text_row + 1) % WIN_ROWS;
    }
    else {
        /* write char */
        wsettextpos(Win_text_row, Win_text_col);

        /* write char's attribute */
        wsetattrpos(Win_text_row, Win_text_col);

        Win_ch[Win_text_row][Win_text_col] = c;
        Win_at[Win_text_row][Win_text_col] = Win_text_attr;

        /* update row and col variables */
        Win_text_col++;

        if (Win_text_col >= DISP_COLS) {
            Win_text_col = 0;
            Win_text_row++;
            if (Win_text_row >= WIN_ROWS) {
                Win_text_row = 0;
            }
        }
    }
}
/*****

/* wputchar c */

/* AND1091 LCD display routine for use with the Multimonitor */
/*****

#include <display.h>
wputchar(r, c, ch)
    int      r, c, ch;
{

    Win_ch[r][c] = ch;
}

```

wputc

wputchar

displib.c

displib.c

```

/*****
/* wputs.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

*/

```

#include <display.h>
wputs(s)
char *s;
{

```

wputs

```

    while (*s) {
        wputc(*s);
        s++;
    }
}

```

```

/*****

```

```

/* wscrollup.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

*/

```

#include <display.h>
wscrollup(n)
int n;
{

```

wscrollup

```

    Win_row_org -= n;
    if (Win_row_org < 0) {
        Win_row_org = 0;
    }
    win_to_nxt();
    dupdate();
}

```

```

/*****

```

```

/* wscrollup.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

```

*/

```

#include <display.h>
wscrollup(n)
int n;
{

```

wscrollup

```

    int lastpos;

    lastpos = WIN_ROWS - Win_row_len - 1;

```

...wscrollup

```

        Win_row_org += n;

        if (Win_row_org > lastpos) {
            Win_row_org = lastpos;
        }

        win_to_nxt();

        dupdate();
    }
}
/*****
/* wsetattrpos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsetattrpos(r, c)
    int      r, c;
{
    Win_text_row = r;
    Win_text_col = c;
}
/*****
/* wsetattr.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsetattr(attr)
    int      attr;
{
    Win_text_attr = attr;
}

/*****
/* wsetcurspos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsetcurspos(r, c)
    int      r, c;
{
    Win_curs_row = r;
    Win_curs_col = c;
}

```

*/

wsetattrpos

*/

wsetattr

*/

wsetcurspos

displib.c

```

} /*****
/* wsettextpos.c
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wsettextpos(- c)
{
    int          r, c;

    Win_text_row = r;
    Win_text_col = c;
} /*****
/* wupdate.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wupdate()
{
    win_to_nxt();
    dupdate();
}

/*****
/* wupdl.c:
/* AND1091 LCD display routine for use with the Multimonitor:
*****/

#include <display.h>
wupdl(i)
{
    int          i;

    int          j;

    if (i < Win_row_org || i >= Win_row_len + Win_row_org)
        return;

    win_to_nln(i);
    j = Win_row_dorg + i - Win_row_org;
    dupdl(j);
}

```

displib.c

...wsetcurspos

*/

wsettextpos

*/

wupdate

*/

wupdl

